ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

GESTIÓN DE UN SISTEMA DE DETECCIÓN DE INTRUSOS EN ENTORNOS DE NUBE PRIVADA

(Management of a intrude detection system in private cloud environments)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Daniel Nicolás Suárez Plata

Octubre – 2019

RESUMEN

La desconfianza en terceros y la falta de control sobre sus propios datos son algunas de las alegaciones que exhiben las entidades corporativas para permanecer en una postura reacia a la hora de migrar a la nube. Por ello, muchas organizaciones optan por trasladar sus servidores o plataformas físicas a una infraestructura de nube propia donde los aspectos relativos a la seguridad y gestión de la nube son responsabilidad exclusiva del propietario.

Se trata de un entorno de nube privada que proporciona acceso a través de Internet, exponiendo los recursos e información almacenada en la nube ante potenciales intentos de vulnerar la seguridad de esta.

Esta solución infiere a las entidades corporativas la necesidad de contar con administradores formados en dichas competencias y herramientas capaces de implementar una arquitectura de seguridad acorde a las exigencias preestablecidas.

El trabajo aborda ambos aspectos en un entorno de nube privada Openstack, donde las herramientas de seguridad desarrolladas hasta el momento centran sus esfuerzos en la protección de la nube en los extremos de la red, obviando el control y el análisis del interior de la misma. Este vacío es cubierto por herramientas de monitorización y generación de alertas como los sistemas de detección de intrusos.

ABSTRACT

Distrust in third parties and lack of control over their own data are some of the reasons that corporations exhibit to remain in a reluctant position regarding cloud migration. Therefore, many organizations choose to move their servers or physical platforms to their own cloud infrastructure where aspects related to cloud security and management are the sole responsibility of the owner.

It is a private cloud environment that provides access through the Internet, exposing the resources and information stored in the cloud to potential attempts to violate its security.

This solution infers to the corporate entities the need to have administrators trained in said competencies and tools capable of implementing a security architecture according to the pre-established requirements.

The paper addresses both aspects in an Openstack private cloud environment, where the security tools developed so far focus their efforts on the protection of the cloud at the ends of the network, obviating control and analysis within the network. This gap is covered by monitoring and alert generation tools such as intrusion detection systems.

Contenido

1 Introducción

El concepto de nube hace referencia a un conjunto de recursos diseñados con un propósito especial. Servicios como el almacenamiento, la capacidad de cómputo y la red, se desarrollan en estos recursos para ser utilizados de forma remota por los clientes. El proveedor de servicios es el propietario de esos recursos y, como tal, se encarga de la administración de aspectos como la seguridad, el acceso y la disponibilidad. Factores como la propiedad y la compartición de los mismos determinan las características del entorno cloud. Un propietario que ofrece a cualquier público sus recursos a cambio de remuneración económica posee lo que se conoce como una infraestructura de nube pública, mientras que, los recursos dedicados exclusivamente a un cliente u organización, forman una infraestructura de nube privada. Entre las grandes ventajas de los entornos de nube destaca la facilidad de acceso a los recursos o información almacenada en estos. Cualquier cliente con conexión a internet y con las credenciales necesarias se encuentra en disposición de acceder a la nube. Esta característica convierte a la nube en una opción muy atractiva para las empresas, y un objetivo suculento para los amigos de lo ajeno.

1.1 Motivación

Cada vez son más las grandes organizaciones que muestran interés por hacer uso de este tipo de entornos. Aunque se encuentren en mayor crecimiento, los servicios de nube pública, los entornos de nube de privada permanecen como la solución más utilizada por las empresas. Esto se debe a que ofrecen un mayor control sobre los parámetros relacionados con la seguridad. No siempre es así, pero por lo general, las empresas suelen implementar su nube privada en el interior de su infraestructura de red, detrás de sus firewalls. El firewall funciona como filtro tanto para el tráfico entrante como saliente, de forma que las empresas pueden seleccionar, en gran medida, el tráfico que circula en el interior de su red y, por consiguiente, alcanzar los recursos de su nube privada. Sin embargo, la nube debe ser accesible no solo para los empleados de la empresa, sino para sus clientes. Esto implica que cierto tipo de tráfico, por mínimo que este sea, circula por la red de la nube privada, es decir, se trata de tráfico que previamente ha sido permitido por el firewall. Bajo este supuesto, se producen situaciones que pueden comprometer la seguridad de la nube. Ataques que tienen como objetivo hacer un sistema o un recurso de red inaccesible para los usuarios, son una de las principales preocupaciones para los administradores. Es el caso del ataque basado en la denegación de servicio, habitual en la capa de red, pero realizable también en las capas de transporte o aplicación.

Los entornos de nube son ideales para realizar este tipo de ataques o también el conocido como fuerza bruta, puesto que permiten la generación de múltiples máquinas ejecutándose simultáneamente. No se trata únicamente de proteger los recursos o información de la empresa ante actividades con origen externo, sino que, la reputación de una empresa envuelta en la realización de algún ataque ya sea por parte de algunos

de sus empleados o de un intruso que ha conseguido establecerse en la nube, podría verse gravemente perjudicada.

Ante este tipo de situaciones aparecen con una relevancia importante los sistemas de detección de intrusos. Herramientas que permiten monitorizar redes en busca de este tipo de actividades, generando notificaciones para que el administrador actúe y tome las medidas que considere oportunas en cada situación.

1.2 Objetivos

La creciente necesidad por parte de las empresas de proteger tanto la infraestructura de su nube como los datos almacenados en ella exige un análisis de las posibles combinaciones de detección y prevención de actividades malintencionadas en entornos de nube privada basados en el software Open Source Openstack.

El proyecto pretende analizar las posibilidades que tiene un administrador de una nube privada para controlar la actividad que tiene lugar en el interior de su red.

La combinación de un sistema de detección de intrusos escuchando en el interior de la nube junto con las herramientas que ofrecen los desarrolladores de Openstack para filtrar el tráfico, permiten realizar múltiples configuraciones en las que prevalece la seguridad y reputación de la empresa en cuestión. Por ello, se pretende implementar situaciones que permitan comprobar el correcto funcionamiento de estas configuraciones concluyendo cuáles son las más adecuadas para un administrador en cada caso.

1.3 Organización del documento

Este documento se estructura de la siguiente forma. Tras este capítulo de introducción, los conceptos teóricos necesarios para seguir el desarrollo del proyecto se presentan en el capítulo 2, principalmente en dos secciones.

La primera sección describe las características básicas de un sistema de detección de intrusos. Se centra en el análisis en profundidad del sistema de detección de intrusos basado en red conocido como Snort.

La segunda sección describe detalladamente las bases del componente de Openstack, Neutron. Se trata del proyecto encargado de gestionar las acciones sobre cualquier red de una nube Openstack, lo que lo convierte en un elemento esencial en este proyecto. Se presenta su diseño desde dos puntos de vista, uno centrado en la distribución de funciones en la infraestructura y otro en la implementación de estas funciones.

A continuación, en el capítulo 3, se discuten las posibilidades de diseño que tiene un administrador de una nube privada a la hora de desplegar, o permitir, desplegar a los tenant (clientes) sus propias redes. Se analizan los dos modelos permitidos por Openstack. Esto permite diseñar una nube privada con el fin de realizar un análisis simulando los posibles escenarios de intrusión presentes en el caso particular que se propone. Se describe el camino de datos en el interior de la nube en cada escenario.

Antes de desarrollar el proceso de implementación, en el capítulo 4 se realiza un análisis en busca de las mejores localizaciones del sistema de detección de intrusos en el interior de la nube privada.

En el capítulo 5, una vez se han descritos los posibles escenarios de intrusión, junto con los posibles emplazamientos del sistema de detección, se procede a reproducir estas situaciones comprobando su efectividad como mecanismo de seguridad en la nube.

Por último, se realiza una conclusión en base a los resultados obtenidos y se proponen líneas de investigación para futuros proyectos.

2 Conceptos teóricos

A continuación, se exponen los principales conceptos teóricos en los que se basa este trabajo, que básicamente son los sistemas de intrusión y los gestores de nube privada.

2.1 Sistemas de detección de intrusos

Un sistema de detección de intrusos es una herramienta utilizada para la monitorización de los eventos que tienen lugar en un sistema. Lo que realmente distingue a un **IDS** (**Intrusion Detection System**) de otras herramientas que ofrecen servicios de monitorización similares, es el posterior análisis que se lleva a cabo sobre estos eventos. Este análisis permite al IDS detectar si se está comprometiendo la seguridad del sistema, alertando al administrador de una posible violación de seguridad. En base al tipo de sistema monitorizado se clasifican los sistemas de detección de intrusos en, **sistemas basados en host** (HIDS) y **sistemas basados en red** (NIDS).

Un **sistema de detección de intrusos basado en el host** realiza la monitorización de todos los componentes de un equipo informático, asegurando la integridad de los datos intercambiados. Para ello supervisa no solo las acciones de los usuarios en base a los privilegios adquiridos por los mismos, sino también a los procesos y archivos. El principal representante de este tipo de sistemas es el bien conocido antivirus, así como sus variantes, como son los antimalware, antispyware, etc.

Un **sistema de detección de intrusos basado en red** monitoriza la red en busca de tráfico que, tras ser analizado, pueda ser clasificado como malicioso, y de esta manera, alertar al administrador para que tome las acciones correspondientes. La combinación de un NIDS junto con un firewall supone la base de cualquier sistema de prevención de intrusos. El NIDS alerta, con carácter preventivo, del tráfico malicioso para que posteriormente sea bloqueado mediante el firewall.

Por otro lado, desde un punto de vista enfocado únicamente en el análisis de eventos, existen dos tipos de sistemas de detección de intrusos. En primer lugar se encuentran los IDS basados en el uso indebido, también conocidos como **IDS basado en firmas**. Se trata de sistemas que analizan los eventos en busca de patrones de ataque conocidos. Estos patrones se encuentran almacenados de alguna forma en el sistema. Por otro lado, existen los **IDS basados en anomalías**. Estos sistemas analizan el comportamiento o las actividades habituales de un usuario con el objetivo de crear un perfil, clasificado como normal. De esta forma, cualquier comportamiento que se aleje de este perfil será detectado como sospechoso.

Los sistemas basados en firmas presentan las siguientes ventajas y desventajas.

Ventajas IDS basado en firmas:

- Capaz de detectar ataques sin crear alertas de falsos positivos, es decir, al realizar el análisis en busca de patrones conocidos, solo se alerta al administrador ante un ataque verdadero.

- Se trata de una herramienta que permite tomar decisiones sobre el tráfico monitorizado fácilmente.

Desventajas IDS basado en firmas:

- No son capaces de detectar nuevos ataques como consecuencia de la detección de ataques sin falsos positivos.
- Por consiguiente, es necesario actualizar estos sistemas ante cada descubrimiento de un nuevo ataque.
- Una pequeña variación de un ataque ya conocido, lo convierte de nuevo, en un ataque indetectable para este tipo de sistemas.

Las ventajas y desventajas relativas a los sistemas basados en anomalías se describen a continuación:

Ventajas IDS basado en anomalías:

- Permiten la detección de nuevos ataques como mejora a los sistemas basados en firmas.
- Al tratarse de sistemas capaces de detectar ataques de los que no se conoce ningún tipo de información, pueden almacenar dicha información para su posterior uso en sistemas basados en firmas.

Desventajas IDS basado en anomalías:

- Debido al comportamiento impredecible de los usuarios, se tiene un gran número de falsos positivos,
- Se trata de sistemas que requieren grandes ciclos de estudio del comportamiento de los usuarios para poder establecer un perfil clasificado como normal.

La seguridad de un sistema reside en la capacidad de mantener la confidencialidad, la integridad y la disponibilidad de este. Por lo tanto, es posible definir la **intrusión** en un sistema, como el intento de comprometer alguna de las bases sobre las que se cimienta la seguridad de este. Ha de tenerse en cuenta que estos intentos pueden venir de usuarios externos, intentando acceder al cloud a través de Internet, usuarios del cloud, intentando hacer uso de privilegios no concedidos, o de usuarios con privilegios, haciendo un mal uso de estos.

Un sistema implementa estas capacidades mediante **firewalls**, mecanismos de **autenticación** y protocolos de **encriptación** entre otros. Sin embargo, en este trabajo tan solo se hace uso de herramientas para el filtrado de tráfico.

Un firewall suele implementarse en los extremos de la red funcionando como filtro para el tráfico entrante y saliente de esta. Esto provoca que los ataques en el interior de la red, es decir, ataques que no son interceptables por el firewall, mantengan amenazada la seguridad de los recursos del sistema. Es en esta situación, donde entra en escena un sistema de detección de intrusos, el cual, como se ha visto, es capaz de detectar tanto

ataques conocidos como desconocidos, actuando como un elemento adicional de prevención en la seguridad del cloud.

La efectividad del IDS está directamente relacionado con factores como el método de detección utilizado, la ubicación dentro de la red o la configuración del IDS [1][2]. Si bien existen alternativas comerciales, el principal representante de este tipo de sistemas es gratuito, y recibe el nombre de Snort.

2.1.1 Snort

Snort es una aplicación Open Source creada en el año 1998 con el objetivo de unificar las características más destacadas de los NIDS comerciales de la época. Se trata de una aplicación que permite la monitorización del tráfico en la red gracias al uso de librerías libcap, utilizadas inicialmente para ofrecer este servicio en la aplicación tcpdump. Además de la monitorización del tráfico, Snort permite su inspección en tiempo real mediante un motor de detección que posee una arquitectura modular basada en plugins. La inspección del tráfico capturado se realiza en busca de indicios que demuestren que se están produciendo actividades maliciosas sobre la red. La búsqueda hace uso de una estructura de detección basada en una lista de reglas flexibles y adaptables a necesidades concretas. Esto, junto con su fácil configuración en un breve intervalo de tiempo, hace que Snort sea uno de los sistemas de detección de intrusos más extendidos dentro de la arquitectura de seguridad de grandes organizaciones.

Puede funcionar tanto como un sistema de monitorización, registrando o sin registrar paquetes, o como un sistema de detección de intrusos completo.

2.1.1.1 Arquitectura de Snort

Snort presenta una arquitectura modular basada en plugins. La mayoría de los módulos de Snort se implementan como plugins, permitiendo a los usuarios realizar una personalización muy detallada del funcionamiento global del sistema.

En un principio, Snort estaba basado en librerías libcap para capturar paquetes. Sin embargo, desde la versión 2.9 se introdujo la librería de adquisición de datos (DAQ). Su función es redirigir los paquetes a Snort, reemplazando las llamadas directas, que Snort realizaba sobre las librerías libcap. Esto permite abstraer la configuración de Snort del hardware o software sobre el que se instale.

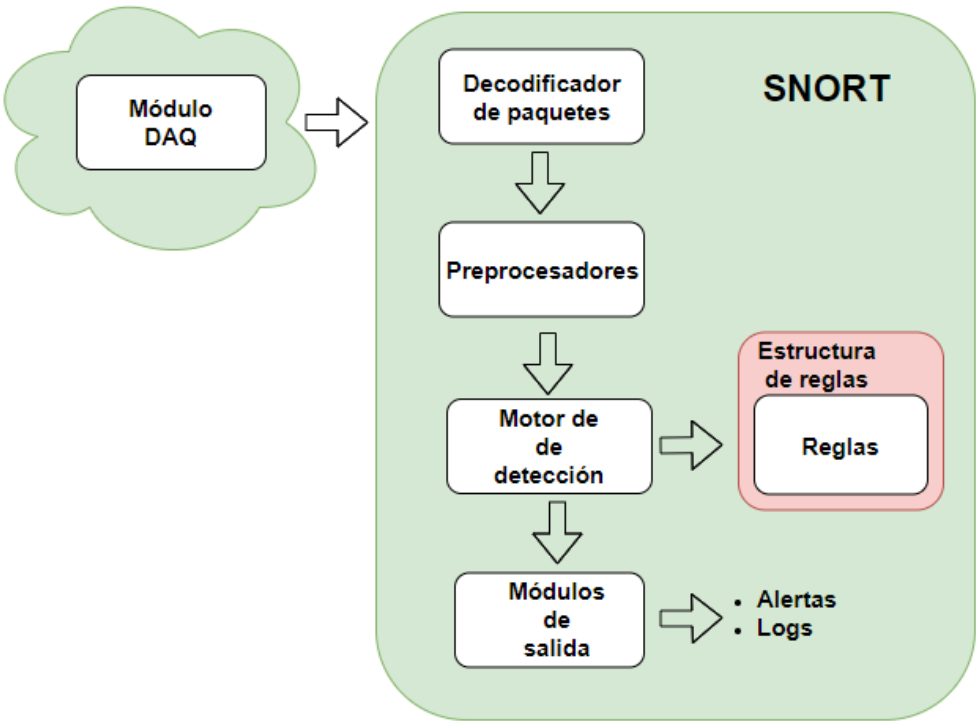

Figura 1. Arquitectura de Snort

Snort está compuesto por cuatro componentes que se muestran en la figura 1. Los paquetes capturados por el módulo DAQ se reenvían directamente a Snort, donde recorren cada uno de los componentes que lo conforman [3].

- **Decodificador de paquetes**
  Se encarga de recibir y procesar los paquetes enviados por el módulo DAQ a Snort. Desgrana la información de cada paquete para almacenarla en estructuras de datos en memoria. Estas estructuras son utilizadas por los preprocesadores y el motor de detección a la hora de realizar el análisis para la detección de intrusos. La decodificación se realiza por capas, empezando por el nivel de enlace y ascendiendo capas en el modelo OSI.
- **Preprocesadores**
  Se trata de un conjunto de plugins, cada uno en busca de un cierto tipo de comportamiento por parte del paquete y que, tras ser determinado, se envía al motor de detección. Su objetivo es permitir el análisis de las estructuras de datos que contienen la información de los paquetes capturados, antes de ser inspeccionadas por el motor de detección. Este análisis previo, permite llevar a cabo determinadas acciones sobre los paquetes, como modificar su contenido o generar alertas directamente desde el preprocesador. Cada plugin está especializado en un tipo de ataque y sin su existencia este no sería detectable en el motor de detección. La presencia de estos elementos se determina en el archivo de configuración de Snort, de manera que, cada paquete procedente del decodificador es enfrentado a todos los plugins activos en el archivo.

- **Motor de detección**
  Se trata del componente central de Snort sobre el que giran el resto de elementos, cuya actuación tiene como consigna principal, aportar más y mejor información al motor de detección. Al tratarse de un sistema de detección de intrusos basado en firmas de ataques conocidos, posee un conjunto de reglas agrupadas en base a tipos de ataque o tipo de aplicación atacada. Estas reglas se actualizan con frecuencia para cubrir los nuevos ataques expuestos. La detección se realiza comparando cada paquete procedente del preprocesador con el conjunto de reglas en busca de coincidencias. En tal caso el paquete es enviado al módulo de salida para que este genere la notificación correspondiente en forma de alerta o log.
- **Módulo de salida**
  Es el módulo encargado de generar la notificación correspondiente, en forma de alerta o log, a un paquete detectado como malicioso en el motor de detección o directamente en un preprocesador.

El flujo del tráfico analizado se controla en base al conjunto de reglas de filtrado establecidas por el administrador.

2.1.1.2 Reglas de Snort

La información relativa a las firmas de ataques conocidos se introduce en cualquier IDS mediante un conjunto de reglas. Se trata del conjunto de instrucciones a partir del cual se genera la estructura de reglas utilizada por el motor de detección durante la monitorización de la red. Las reglas permiten adaptar cualquier IDS a las necesidades concretas de un usuario.

Una regla de Snort está compuesta por dos partes, una cabecera y un conjunto de opciones.

La cabecera contiene una serie de elementos con información destinada a identificar aquellos paquetes sobre los que tiene alcance la regla. Además, contiene la información relativa a la acción que se ha de ejecutar sobre estos paquetes. La estructura de la cabecera se describe a continuación:

<acción> <protocolo> <red origen> <puerto origen> <dirección> <red destino><puerto destino>

El campo **acción** es configurable con siete tipos de acciones:

- alert: genera una alerta y después registra el paquete.
- log: únicamente registra el paquete.
- pass: ignora el paquete.
- activate: genera una alerta y activa otra regla (regla dinámica).
- dynamic: permanece inactiva hasta que una regla de tipo activate la activa. Una vez activada funciona como una regla de tipo log.
- drop: bloquea el paquete y lo registra como si fuera una regla log.

- reject: bloquea el paquete, lo registra y envía un paquete RST, si se trata de un protocolo TCP, o un mensaje de puerto inalcanzable ICMP, si se trata de un protocolo UDP.
- sdrop: bloquea el paquete, pero no lo registra.

El siguiente campo describe el **protocolo** utilizado por aquellos paquetes que serán sometidos a un análisis en Snort. Se detectan hasta cuatro protocolos en Snort: TCP, UDP, IP e ICMP.

Los campos **red** y **puerto origen** detallan el socket origen con el que cualquier paquete será comparado al decidir si se debe o no realizar una acción sobre él.

Del mismo modo los campos de **red** y **puerto destino** describen el socket destino sobre el que se compara un paquete antes de ser analizado por el IDS.

El campo **dirección** se utiliza para configurar el sentido de la comunicación a ser analizada. Es configurable con un único sentido → o con una comunicación bidireccional <>.

Es condición necesaria que la cabecera de un paquete coincida con todos estos campos para que se proceda a realizar un análisis detallado del paquete. El análisis ejecutado por el motor de detección se basa en la sección de opciones de la regla.

Las opciones de una regla especifican qué información debe ser analizada en el paquete antes de ejecutar la acción declarada en la cabecera de la regla. También contiene el mensaje del log o de las alertas. Se clasifican en cuatro categorías:

- Generales: msg, reference, gid, sid o rev. Se trata de opciones que ofrecen información acerca de la regla, pero no tienen efecto en el proceso de detección.
- Payload: content, urilen, isdataat o pcre. Se trata de opciones que proporcionan información utilizada para la detección de ataques y en particular información relativa al payload del paquete a detectar. La opción content especifica el patrón buscado por el motor de detección, siendo un campo importantísimo en la detección de ataques.
- Non-payload: fragoffset, ttl, tos, id, ipopts o fragbits. Sirven para especificar patrones en otros campos del paquete que no sean el payload.
- Postdetección: logto o activates. Permiten activar reglas específicas que se ejecutan al terminar de hacerlo la actual regla.

Snort utiliza las reglas incluidas en el fichero de configuración, durante la primera fase de ejecución para generar la estructura de detección [3][4].

2.1.1.3 Modos de ejecución

Como se ha mencionado, Snort es un sistema de detección y prevención de intrusos que puede trabajar en cualquiera de los siguientes modos:

- **Modo Sniffer**: escucha todo el tráfico de la red, y dependiendo de la configuración, muestra el resultado por pantalla. El resultado puede estar

compuesto por cabeceras TCP,UDP o ICMP, los datos del paquete o una combinación de ambos. Se activa ejecutando Snort junto con el parámetro -v: *./snort -v o ./snort -vde* para visualizar información más detallada. A diferencia de otros modos, no registra los resultados mostrados en pantalla en la memoria del sistema.

- **Modo Packet Logger**: se trata de una extensión del modo sniffer, en la que además de capturar todos los paquetes entrantes y salientes de la red, se procede a almacenarlos de forma persistente en el disco duro. La ubicación donde se almacenan los resultados se especifica en el comando de ejecución de Snort mediante el parámetro -l: *./snort -l /home/user /logs/*. Para visualizar estos datos almacenados, Snort permite utilizar la opción -r durante la ejecución del comando. Esta opción permite filtrar por palabras clave los ficheros que almacenan las capturas.
- **Modo NIDS**: se trata del modo más complejo y completo en el que puede funcionar Snort. Permite la captura y manipulación de los paquetes entrantes y salientes del segmento de red que se está monitorizando. Utiliza un conjunto de reglas incluidas en el fichero de configuración con las que comparar los paquetes capturados. Si las características del paquete y la regla coinciden, se analiza la información del paquete indicada por la regla en busca de actividades maliciosas.

  En el proceso de ejecución de este modo se distinguen dos fases.

  - Una primera fase en la que se lee del fichero de configuración de Snort, se analiza el conjunto de reglas, se configuran los componentes y se crea la estructura de detección.
  - En la segunda se reciben los paquetes de la red a través del módulo DAQ, se decodifican y se envían al preprocesador, para que posteriormente se redirijan al motor de detección de Snort, que será el encargado de realizar el análisis definitivo.

  La ejecución se realiza con el parámetro -d y -c para recibir como parámetro de entrada el fichero de configuración de Snort, necesario para generar la estructura de detección y configurar todos los componentes: *./snort -d -c /etc/snort/snort.conf*.

- **Modo Inline**: se puede decir que, en este modo, Snort funciona como un sistema de prevención de intrusos, ya que, a la funcionalidad de sistema detector de intrusos se le incorpora la posibilidad de interaccionar con un firewall, de modo que, este sea el encargado de regular el tráfico marcado por el NIDS.
  Entre las reglas que implementan esta funcionalidad, destacan aquellas que utilizan como acción alguna de las siguientes configuraciones: drop, reject o sdrop. Cuando se analice un paquete en el que se detecte algún patrón descrito por una regla, se enviará una petición a iptables para que actúe sobre este.
  Para poder usar este modo es necesario configurar el sistema en modo bridge y utilizar algún módulo de salida especial que comunique Snort con Iptables [5][6].

2.2 Sistemas operativos de Cloud: Openstack

Un sistema operativo de nube es un tipo de sistema diseñado para operar en entornos de computación en nube y virtualización. Un sistema operativo en la nube gestiona el funcionamiento, la ejecución y los procesos de las máquinas virtuales, los servidores virtuales y la infraestructura virtual, así como los recursos de hardware y software de back-end.

Un sistema operativo en la nube gestiona principalmente la operación de una o más máquinas virtuales dentro de un entorno virtualizado. Dependiendo del entorno virtual y los servicios en la nube en uso, la funcionalidad de los sistemas operativos en la nube varía.

Por ejemplo, un sistema operativo en la nube desarrollado para ser utilizado dentro de un entorno informático específico administrará los procesos y subprocesos de una sola máquina o de una agrupación de máquinas virtuales y servidores. De manera similar, un sistema operativo en la nube liviano podría proporcionar a los usuarios finales aplicaciones y servicios preinstalados, a los que se accede a través de un navegador de Internet.

Entre las soluciones comerciales existentes cabe destacar Microsoft Windows Azure y Google Chrome OS, mientras que entre las soluciones Opensource la más utilizada es OpenStack.

OpenStack es un sistema operativo en nube que controla grandes grupos de recursos de computación, almacenamiento y redes a través de un centro de datos, todo ello gestionado a través de un panel de control que proporciona a los administradores el control al tiempo que permite a sus usuarios aprovisionar recursos a través de una interfaz web. Desde el punto de vista del despliegue, Openstack ofrece un modelo de "infraestructura como servicio" (IaaS), por medio del cual se ponen a disposición de los clientes servidores y redes virtuales.

OpenStack está a su vez compuesto por una serie de proyectos de software libre, de los cuales ahora se indican únicamente los elementos utilizados en este. En concreto, los proyectos utilizados son:

- **Keystone**: Proporciona autenticación de cliente de acceso a la API (Application Program Interface), además de la detección de servicios de identidad. Soporta protocolos relacoinados con dicha tarea, como son LDAP, OAuth, OpenID Connect, SAML y SQL.
- **Neutron**: Proporciona conectividad de red como servicio a otros proyectos de OpenStack, como por ejemplo Nova, el cual utiliza la API de Neutron para solicitar la conexión de las máquinas virtuales a la red determinada. Permite la creación de redes, subredes, enrutadores, cortafuegos, y redes privadas virtuales. Cuenta con una arquitectura modular que se integra con múltiples tecnologías de proveedores de red externos.

- **Glance**: Almacena y gestiona imágenes de los discos de las máquinas virtuales.
- **Nova**: Proporciona una forma de aprovisionar servidores virtuales. Nova soporta la creación de máquinas virtuales y servidores baremetal. Nova se ejecuta como un conjunto de daemons encima de los servidores Linux existentes para proporcionar ese servicio. Para el uso de nova es necesario que los tres servicios anteriores estén previamente activos:
- **Horizon**: Es la implementación del panel de control de OpenStack, que es extensible y proporciona una interfaz de usuario basada en web para los servicios de OpenStack.
- **Cinder**: Es un servicio de almacenamiento en bloque para OpenStack.. Virtualiza la gestión de dispositivos de almacenamiento y proporciona a los usuarios finales una API de autoservicio para solicitar y consumir esos recursos, sin necesidad de saber dónde se implementa realmente su almacenamiento o en qué tipo de dispositivo se lleva a cabo.

Con tal cantidad de componentes, el proceso de instalación de OpenStack puede ser bastante complejo, por lo que existe, además de abundante información técnica, multitud de tutoriales y ejemplos de configuración de diferentes soluciones basadas en OpenStack, así como implementaciones desarrolladas a medida por algunas empresas, y que son ofrecidas a la comunidad añadiendo ofertas de mantenimiento.

2.2.1 Evolución del servicio de red en Openstack

En las primeras versiones de Openstack el servicio de red era un subcomponente del servicio compute (nova) conocido como nova-network. Este servicio es simple y aborda las diferentes necesidades del usuario mediante tres tipos de lo que se definen como administradores de red, **FlatManager**, **DhcpManager** y **VlanManager**.

El administrador de red es el elemento encargado de ofrecer una topología de red a la implementación de Openstack en la que se configure.

Tanto FlatManager como DhcpManager basan su topología de red en un único bridge virtual para cada nodo de cómputo, al que se conectan todas las máquinas virtuales generadas por Openstack en dicho nodo, como se ve en la figura 2. Es importante tener en cuenta que Nova-Network se basa en la tecnología Linux Bridge para la creación de bridges virtuales, ya que se trata de una tecnología con una importante limitación en lo que se refiere a la virtualización de la red. Solo es posible conectar un único Linux Bridge a la interfaz física del equipo. Este problema podría solventarse al añadir un equipo vlan entre cada Linux Bridge y la interfaz física del equipo actuando como subinterfaces con etiquetas vlan. Sin embargo, esto no es posible debido a que ni FlatManager ni DhcpManager soportan la tecnología vlan.

Esta limitación hace imposible para el proveedor del cloud, aislar completamente la red de cada tenant dentro de un mismo nodo. Todas las instancias de un nodo comparten

el mismo espacio de direcciones, por lo que serán visibles para el resto independientemente de quién sea el propietario de la instancia.

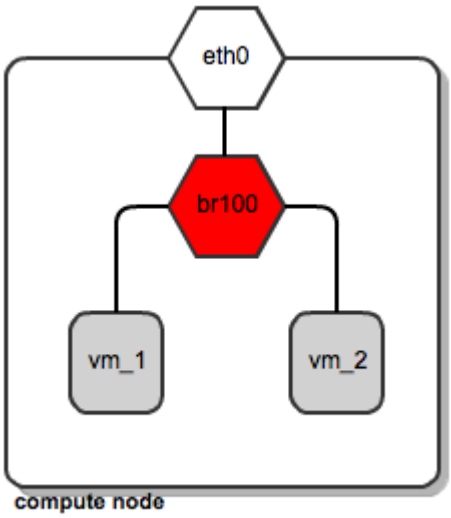

Figura 2. Topología de red FlatManager **[7].**

Para ofrecer un mínimo grado de seguridad a los usuarios es posible configurar el administrador de red para que no permita ningún tráfico entrante a las instancias. Esto se consigue mediante los conocidos grupos de seguridad de Openstack implementados por la utilidad nativa de Linux iptables en el Linux Bridge.

Mientras que DhcpManager es una pequeña extensión de FlatManager, VlanManager intenta resolver los dos principales problemas de estas configuraciones: la falta de escalabilidad y la falta de aislamiento entre tenants. Para ello hace uso de redes vlan, cuyo propósito es separar en varios dominios una red física, de forma que, instancias en distintos dominios no sean visibles entre sí.

VlanManager configura la red creando un bridge dedicado a cada tenant en cada nodo y adjunta a la interfaz física del nodo una interfaz vlan por cada bridge que se conecte.

De este modo, todas las instancias en un mismo nodo no tendrán que utilizar el mismo grupo de direcciones, aparecen las subredes y se consigue evitar la visibilidad entre instancias de distintos propietarios. Además, es posible situar instancias de un mismo tenant en nodos distintos, aumentando la escalabilidad de la nube.

La figura 3 muestra una topología de red ofrecida por el servicio nova-network utilizando el administrador de red VlanManager. Vemos que la comunicación entre instancias del mismo tenant situadas en distintos nodos, es posible gracias a la virtualización vlan. Cualquier paquete de datos que atraviesa una subinterfaz vlan con dirección al exterior del nodo, quedará identificado con una etiqueta vlan única para el tenant. Y cualquier paquete que atraviese la interfaz física con destino a una máquina virtual del nodo, será reenviado por la subinterfaz vlan que identifique su etiqueta.

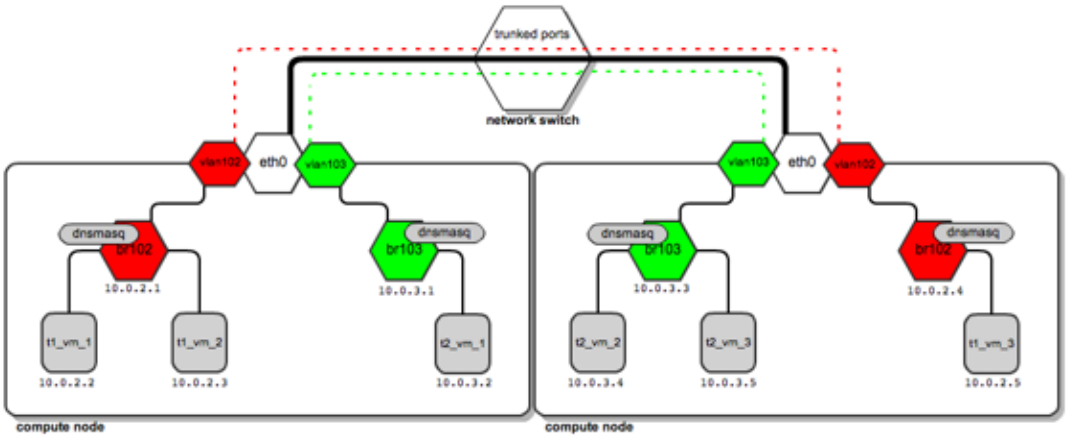

Figura 3. Topología de red VlanManager [7].

A pesar de las mejoras ofrecidas por VlanManager, continúan existiendo importantes limitaciones para los proveedores, en especial permanecen dos grandes problemas.

Como se ha visto antes, la introducción de un bridge virtual dedicado para cada tenant permite utilizar un grupo de direcciones distinto para cada dominio dentro del mismo nodo. Sin embargo, esto sigue sin permitir que tenants distintos utilicen el mismo esquema de direccionamiento en dominios diferentes, es decir, dos tenant con dominios separados por vlan no pueden direccionar sus instancias con la misma ip.

Por otro lado, el campo para la etiqueta vlan dispone tan solo de 12 bits, permitiendo un total de 4096 etiquetas distintas y, por lo tanto, limitando el número de tenants en el cloud.

La evolución del servicio de red continuó gracias al enfoque que los desarrolladores de Openstack impusieron para ofrecer cada servicio como un proyecto independiente. De esta manera se simplifico la complejidad de cada proyecto permitiendo a los desarrolladores con interés y experiencia en un servicio particular de Openstack centrarse en él.

El proyecto encargado de ofrecer el servicio de red se conoció en un principio como Quantum para terminar siendo sustituido por Neutron.

Neutron abordó las deficiencias en las tecnologías de red basadas en un back-end integrable como Quantum y permitió tener un mayor control en los entornos cloud multi tenant. Se diseñó para poder configurar dicho back-end mediante un plugin que permitiese a los operadores de red ofrecer diferentes tecnologías para la virtualización de red. Cada tenant dispone de la posibilidad de utilizar el plugin que mejor se adapte a sus necesidades para crear múltiples redes privadas y administrar el direccionamiento directamente.

Además, existen extensiones de Neutron que complementan estos servicios de red básicos. Entre los servicios que ofrecen estas extensiones se incluyen, un control adicional sobre las políticas de seguridad, la posibilidad de ofrecer calidad del servicio

distinguiendo tráficos, monitorizar y resolver problemas, así como la capacidad de implementar servicios de red avanzados, como firewall as a service o VPN as a service [7][8].

2.2.2 Servicio de red (Neutron)

La virtualización de la red surge ante el importante crecimiento de la virtualización de los recursos informáticos. Estos necesitan de redes a las que conectarse para poder enviar y recibir datos. Los desarrolladores de Openstack son conscientes de estas necesidades y se aprovechan del potencial que tienen las redes definidas por software para ofrecer un entorno de virtualización mucho más completo que el utilizado hasta el momento. Es por esto por lo que surge Neutron, un proyecto que define la red mediante software (SDN). Actualmente, Openstack utiliza Neutron para entregar la red como un servicio independiente, permitiendo integrar equipos de red como routers, switches y firewalls junto a otros equipos informáticos en un mismo entorno de virtualización.

Puesto que se trata del actual servicio de red ofrecido en Openstack se hará un análisis detallado de los servicios o agentes involucrados, los componentes que implementan la API, su localización dentro de la infraestructura física de una implementación de Openstack, los modelos de entregar el servicio y las opciones que existen a la hora de seleccionar un plugin para el Neutron-L2-plugin-Agent.

2.2.2.1 Servicios y agentes de Neutron

Neutron utiliza un back-end configurable para dar soporte a los servicios de red con las mejoras consecuentes que ello supone respecto a tecnologías de red previas a Quantum. Además, se trata de un componente cuyos servicios pueden ser ejecutados en un nodo dedicado o continuar con una arquitectura similar a la que se venía utilizando con nova-network, compartiendo nodo con otros componentes como nova.

Sin embargo, en este caso, la compartición de recursos no implica limitaciones como las sufridas en el servicio nova-network. Se desaprovecha gran parte del potencial que ofrece Neutron. Aparecen limitaciones relacionadas con la sobresaturación de recursos como la CPU, perjudicando el rendimiento de la nube.

Neutron cuenta con dos tipos de servicios distribuidos en una infraestructura multinodo: el neutron-server y los agentes.

- **Neutron**-server: se trata del servicio que ejecuta el Openstack Networking API server. Expone de esta manera la API de Neutron a los usuarios del cloud, permitiéndoles el acceso a todos los servicios de red a la hora de diseñar topologías de red adecuadas para sus necesidades particulares. Actúa como un controlador central para el resto de los agentes.
- **Neutron-L2-plugin-agent**: este agente se ejecuta en cada hipervisor para llevar a cabo la configuración de la tecnología encargada de virtualizar la red. Depende del plugin seleccionado para la implementación y define la tecnología que utilizará Neutron para virtualizar la red. El agente es el proceso que se ejecuta

en cada nodo, encargado de hacer llegar al plugin de cada hipervisor las acciones que debe llevar a cabo.

- **Neutron-L3-agent**: es el agente encargado de ofrecer servicios de enrutamiento para permitir las comunicaciones entre redes tenant. Ofrece el servicio de Network Address Translation (NAT) para permitir las comunicaciones entre instancias del cloud e internet. Concretamente realiza SNAT en los paquetes salientes del cloud cambiándoles la dirección de origen privada por la dirección ip pública del router. Para los paquetes entrantes realiza un procedimiento similar con la dirección destino, sustituyendo la dirección ip flotante de la instancia por su dirección privada mediante lo que se conoce como DNAT. Se trata de un agente opcional que depende del plugin, pero veremos que tanto OpenvSwitch como Linux Bridge lo utilizan.
- **Neutron-dhcp-agent**: se encarga de ofrecer el servicio dhcp en las redes privadas de los tenants. Al igual que Neutron-L3-agent es un agente opcional y depende del plugin instalado.

La comunicación entre estos servicios se realiza mediante el reenvío de las peticiones que realizan los usuarios a través de la API de Neutron. Estos comandos son recibidos por el Neutron-server, que se comunica con el agente correspondiente mediante llamadas a procedimientos remotos (RPC). Algunos de estos agentes, dependiendo del plugin instalado, utilizan un pequeño almacenamiento temporal para no perder estas llamadas, a esta técnica se la conoce como message queuing [8][9].

2.2.2.2 Plugins para Neutron-L2-plugin-Agent

Este agente se ejecuta en cada hipervisor con el objetivo de configurar un switch virtual en cada nodo. Esta configuración depende complemente del plugin que se decida instalar. Dentro del contexto de las redes basadas en un back-end integrable, como lo son las redes creadas por Neutron, un plugin no es más que el conjunto de tecnologías que se deciden utilizar para implementar las peticiones a la API de Neutron. De modo que, algunos plugins de Openstack Networking pueden usar herramientas de virtualización nativas de Linux como Linux Bridge o iptables, mientras que otros pueden usar tecnologías más avanzadas, como túneles virtuales o OpenFlow, para proporcionar beneficios similares.

Al ser un agente cuyo mapeo de funciones depende completamente del plugin seleccionado, es necesario analizar brevemente las características más destacadas de algunos de estos. Entre los seleccionados más habituales se encuentran: Linux Bridge, OpenvSwitch y Modular Layer2.

Linux Bridge

Se trata de un software incluido en el Kernel de Linux para la creación de bridges virtuales en equipos corriendo alguna distribución GNU/LINUX. Se trata de un equipo virtual que emula el funcionamiento de un bridge físico estándar. Un Linux Bridge por sí solo no puede transmitir ni recibir hasta que no se mapee sobre un dispositivo físico. Consta de los siguientes componentes:

- Puertos de red (o interfaces): se utiliza para reenviar el tráfico entre bridges o a otros hosts en la red.
- Un plano de control: se utiliza para ejecutar el spanning tree protocolo (STP) que calcula el árbol de expansión mínimo, evitando que los bucles bloqueen la red.
- Un plano de reenvío: se utiliza para procesar tramas entrantes desde los puertos y reenviarlas al puerto adecuado basando su decisión en la base de datos de aprendizaje MAC.
- Base de datos de aprendizaje MAC: se utiliza para realizar un seguimiento de las ubicaciones de cada host en la LAN.

Las decisiones de reenvío se basan en tablas de direcciones MAC construidas mediante el aprendizaje de hosts conectados a cada puerto. Si no se puede encontrar una entrada para una dirección MAC dada, se transmitirá la trama a todos los puertos excepto el que recibió el mensaje [10][11].

OpenvSwitch

OpenvSwitch es un software incluido a partir de la versión 3.3 del Kernel de Linux para la creación de bridges virtuales multicapa. Su introducción mejora el Linux Bridge de una única capa. Puede funcionar como un switch virtual comportándose como un switch físico o integrarse en un dispositivo físico funcionando como plano de control. Permite conectar a sus puertos tanto interfaces TAP de instancias como espacios de nombre. Además, permite utilizar tecnologías para la virtualización mejores que VLAN como VXLAN o GRE [12].

Un bridge OpenvSwitch puede funcionar en modo "normal" y en modo "flujo".

- En modo normal, actúa como un switch físico estándar de capa de enlace basado en tablas de direcciones MAC. Para cada trama entrante, aprende su dirección MAC de origen y la asocia al puerto de entrada. Para reenviar la trama existen dos posibilidades. Si la dirección Mac destino ha sido aprendida previamente, envía la trama al puerto correspondiente. Si no ha sido aprendida se envía por todos los puertos del bridge exceptuando el puerto entrante. Las tramas de difusión y multidifusión se envían como suele ser habitual.
- En modo flujo, el bridge utiliza la tabla de flujo con cada trama entrante. Los flujos son reglas que pueden ser configuradas e instaladas en el bridge usando el protocolo OpenFlow con la herramienta ovs-ofctl add-flow. Cada regla de flujo está compuesta por una parte que establece la coincidencia a encontrar en la trama entrante y otra con la acción a ejecutar sobre la trama. Aquellas reglas de flujo configuradas para llevar acabo la acción "NORMAL", consultan la tabla de direcciones MAC del bridge para realizar la acción adecuada en base a esta.

OpenFlow es un protocolo de red programable en dispositivos hardware. Surge como solución a los problemas de interacción entre dispositivos hardware de distintos fabricantes. Sirve para configurar y administrar dispositivos de red. Permite abstraer el plano de control de un conjunto de switches en un controlador central. De esta forma un administrador puede definir el comportamiento de la red a través de un software en el controlador.

Cualquier red virtualizada mediante un software, como OpenvSwitch, basado en OpenFlow consta de tres componentes: el controlador central, el protocolo OpenFlow y un conjunto de tablas de flujo almacenadas en cada switch de la red.

La figura 4 muestra la arquitectura de una red creada mediante un software basado en OpenFlow.

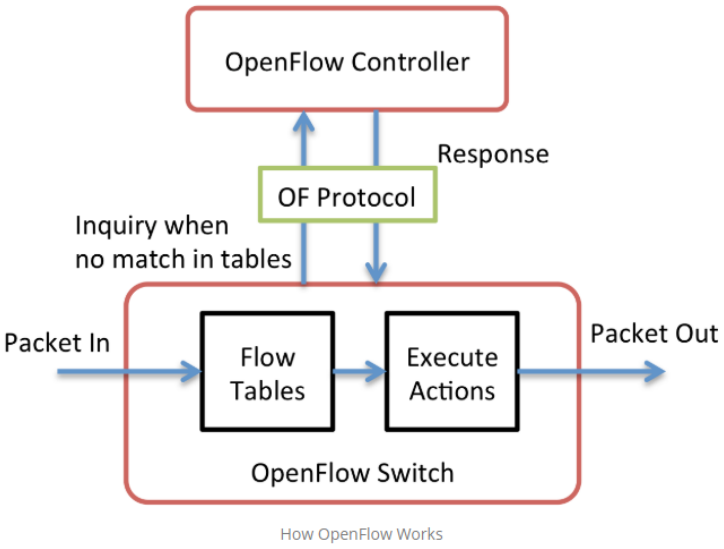

Figura 4. Arquitectura de una red basada en OpenFlow **[13].**

- El protocolo OpenFlow funciona como un mecanismo de comunicación seguro entre el controlador y el switch OpenFlow.
- El controlador puede crear o eliminar reglas para establecer el comportamiento de cada equipo bajo su administración a la hora de reenviar datos.
- La tabla de flujos almacena una colección de reglas para el tráfico entrante.

Cada trama, paquete, o segmento entrante en el switch OpenFlow será comparado con las reglas de la tabla de flujo en busca de alguna entrada que coincida. En caso de existir alguna entrada en la tabla, se procede a aplicar la acción establecida en dicha regla de fujo. En caso de no existir tal entrada, se envía una petición al controlador con el objetivo de obtener una nueva regla de flujo. Una vez se procese la respuesta, el switch aplicará la acción correspondiente a la nueva regla. Mientras tanto el paquete permanece en espera.

El conjunto de componentes que forman OpenvSwitch se muestra en la figura 5. Se diferencia entre el espacio de Kernel y el espacio de usuario. El espacio de Kernel, compuesto por un módulo kernel, es el lugar donde se lleva a cabo la conmutación, mantiene una pequeña memoria cache con algunas de las reglas de la tabla de flujo con el fin de agilizar la conmutación. El espacio de usuario es el encargado de procesar las entradas que no han encontrado coincidencia en el almacenamiento cache del módulo Kernel, por ello, las entradas que atraviesan este espacio sufren un ligero retraso en el proceso de conmutación. Está compuesto por un daemon, ovs-vswitchd, y una base de datos, ovsdb, de carácter persistente para almacenar información relevante de los switches. Además, existe un controlador, mediante el cual es posible definir la red por software, y un conjunto de interfaces de línea de comandos, como ovs-dpctl, ovs-vsctl, ovs-appctl y ovs-ofctl, para la administración y gestión del resto de componentes.

La comunicación entre componentes se realiza mediante protocolos como OVSDB, utilizado para gestionar la base de datos, y Netlink, encargado de la comunicación entre los espacios de usuario y kernel. La comunicación entre los usuarios y los componentes se realiza con las herramientas CLI que hemos visto:

- Ovs-vsctl permite un control externo de ovs-vswitchd. Esta interfaz tiene acceso al kernel, a través de Netlink, y también al servidor ovsdb. Su propósito es administrar múltiples bridges y está involucrado en la conmutación de datos. Es un componente central del sistema.
- Ovs-dpctl permite la configuración externa del módulo Kernel.
- Ovs-appctl y ovs-ofctl permiten la comunicación directa con el Ovs-vswitchd. El primero envía comandos con el fin de arrancar daemons de OpenvSwitch, mientras que el segundo permite administrar las tablas de flujo de los bridges utilizando el protocolo OpenFlow.
- Ovs-pki es la herramienta que permite la creación y administración de una infraestructura pki

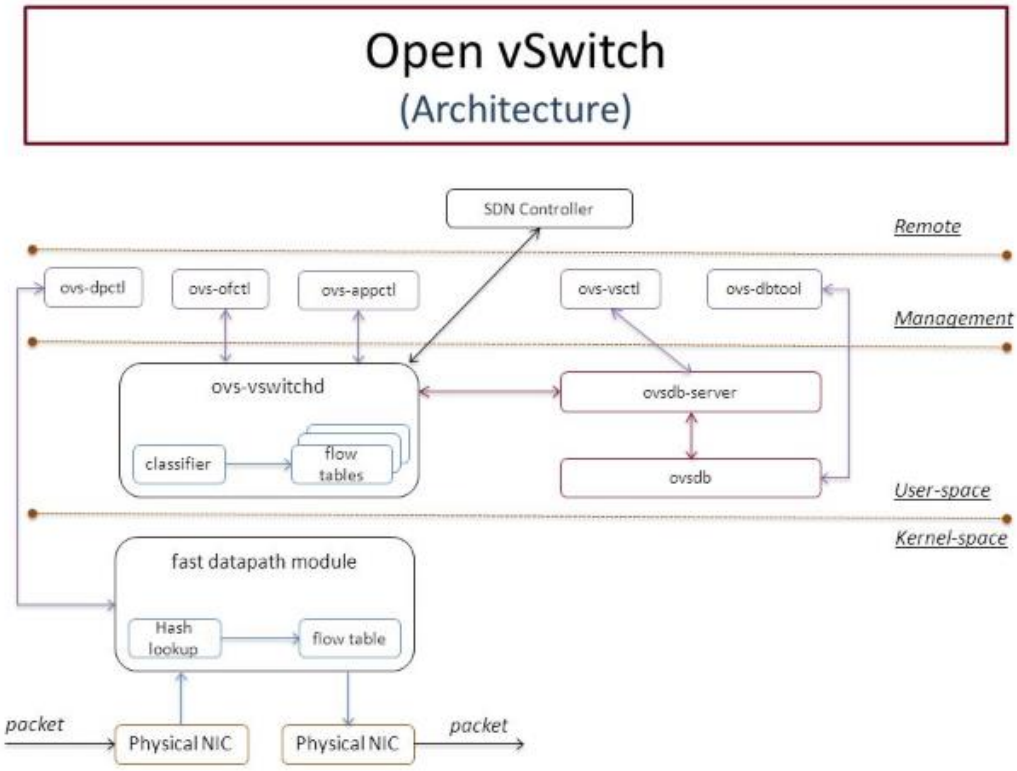

Figura 5. Arquitectura de OpenvSwitch **[13].**

Aunque únicamente se ha expuesto el principal servicio de OpenvSwitch, como es el control del tráfico, no es el único. Otros servicios básicos que ofrece OpenvSwitch son la seguridad, la monitorización del tráfico y la calidad de servicio.

- Filtrado de paquetes. Utilizado como mecanismo de seguridad para los usuarios permitiendo seleccionar sus tráficos.
- La monitorización del tráfico se lleva a cabo a través de protocolos como NetFlow, sFlow, SPAN o RSPAN.
- Soporte de técnicas para la calidad de servicio como Traffic Queuing y Traffic Shaping.
- El control del tráfico, basado en reglas de flujo, se realiza mediante el protocolo de comunicaciones Open Flow y otros protocolos para la administración de bases de datos como OVSDB.

Modular Layer 2

Este plugin no es una nueva tecnología como tal, sino que más bien, se puede definir como una mejora que integra los plugin anteriores. Se introdujo en las últimas versiones de Openstack para dar soporte a varias tecnologías de red simultáneamente, permitiendo utilizar varios tipos de segmentación concurrentemente. Los plugins como OpenvSwitch o Linux Bridge se integran como mecanismos en los archivos de configuración de modular layer2 Anteriormente, implementaciones de red basadas en un plugin especifico solo

podían utilizar dicho plugin configurado desde la instalación. Es por esto, por lo que este nuevo plugin resulta de gran utilidad, integrando en una única infraestructura redes implementadas desde su instalación con un plugin específico. Una implementación de Openstack utilizando el plugin modular layer2 permite tener nodos ejecutando distintos plugins, por ejemplo, un nodo virtualizando la red mediante OpenvSwitch y otro, realizando la misma función, mediante Linux Bridge. Este mismo caso es posible dentro de un mismo nodo físico [8][12].

2.2.2.3 Arquitectura de Neutron

Neutron distribuye y ejecuta sus servicios en distintos nodos de Openstack con el objetivo de lograr un servicio de red distribuido funcionando al máximo de su rendimiento.

Una vez se encuentran expuestos los servicios y agentes envueltos en el proyecto de Neutron, se describe su integración dentro de la infraestructura de la nube.

Una infraestructura de Openstack sin grandes restricciones debe contar con al menos tres nodos físicos albergando los distintos servicios de red. La figura 6 muestra la arquitectura de red para una infraestructura de Openstack con los siguientes nodos: nodo controlador, nodo de red y nodo de computo.

El **nodo controlador** se encarga de ejecutar y exponer a los usuarios las APIs de todos los componentes de Openstack, entre ellas la API de Neutron, es decir, en este nodo se encuentra corriendo el servicio Neutron-server. Se encuentra situado entre dos redes físicas, la red para las llamadas a la API y la red de gestión.

El **nodo de red** alberga la mayor parte de servicios y agentes de red como es de esperar. Es un nodo dedicado exclusivamente al servicio de red. Los agentes como Neutron-L2-plugin-Agent, Neutron-L3-Agent, Neutron-Dhcp-Agent y Neutron-Metering-Agent se ejecutan en este nodo. Cada uno de estos agentes se encuentra en disposición de atender una llamada a procedimiento remoto o RPC procedente del nodo controlador y a continuación procesar la petición. Además de estar conectado a la red de gestión por donde se transportan estas llamadas, se encuentra en contacto directo con la red de datos y la red externa. Debido a que es el único nodo en el que se ejecuta el Neutron-L3-Agent y por tanto el único nodo capaz de llevar a cabo funciones de red, como el enrutamiento o el servicio NAT, cualquier comunicación con internet debe atravesarlo.

Por último, esta infraestructura cuenta con uno o varios **nodos de computo** corriendo el Neutron-L2-plugin-Agent. Este agente se encarga de la configuración del plugin en cada nodo, que como veremos, no tiene por qué ser el mismo en todos gracias al plugin Modular Layer2. Este nodo se encuentra entre dos redes físicas como son la red de gestión y la red de datos.

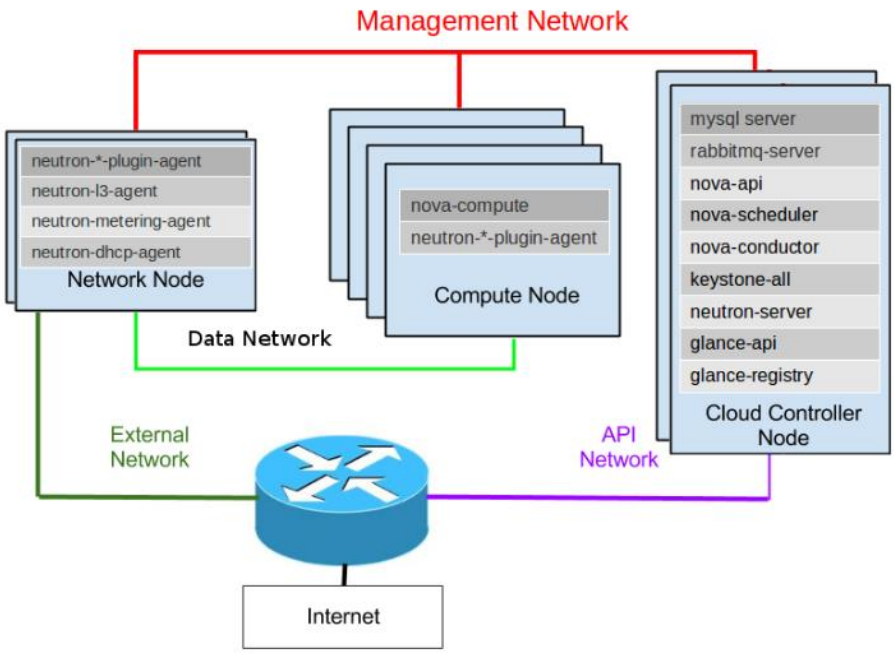

Figura 6. Arquitectura de Neutron **[8].**

Con esta arquitectura de red sobre esta infraestructura física, quedan definidas cuatro redes físicas con cuatro objetivos bien distinguidos. Estas redes conectan entre si los nodos de Openstack y son utilizadas por los servicios de Neutron para cubrir distintos ámbitos como la gestión, el acceso a la API, la comunicación externa e interna.

- **Red de datos**: ofrece soporte para cualquier comunicación en la que intervengan instancias del cloud. Será utilizada por los tenant para acceder a las instancias. Conecta los nodos de cómputo y el nodo de red ya que este último se encarga de enrutar las comunicaciones.
- **Red de gestión**: ofrece soporte para que los servicios de Openstack efectúan las operaciones necesarias como reacción a las llamadas a la API. En el caso del servicio de red, transporta las llamadas a procedimientos remotos que efectúa el servicio Neutron-Server al resto de agentes. Por tanto, esta red conecta el nodo controlador con el resto de los nodos en los que se ejecute algún agente.
- **Red API**: expone tanto para los administradores del cloud como para los administradores de cada tenant las APIs de cualquier componente de Openstack. Se trata de una red conectada al nodo controlador y completamente accesible desde internet para cualquier persona con las credenciales necesarias.
- **Red externa**: proporciona acceso desde internet a las máquinas virtuales situadas en el interior del cloud. Tiene acceso directo al nodo de red, ya que como se mencionó antes, cualquier comunicación entre el interior y el exterior del cloud debe pasar por este nodo.

En caso de no contar con los recursos suficientes o de no tener unas exigencias altísimas en cuanto a recursos de memoria o CPU, es posible implementar esta arquitectura sobre

una infraestructura con un menor número de nodos. Una opción es eliminar el nodo controlador e integrar los servicios que se ejecutan en este junto a los servicios del nodo de red en un único nodo. Otra opción existente pero que tan solo se utiliza para crear entornos de prueba es la conocida como todo en uno. Esta opción consiste en instalar los servicios básicos de Openstack sobre un único equipo.

Los servicios de red no solo interaccionan entre sí. Otros componentes de Openstack hacen uso de estos servicios o Neutron necesita de estos componentes. En los siguientes servicios de Openstack se produce de una u otra forma alguna interacción con Neutron:

- **Keystone**: cualquier usuario debe introducir sus credenciales para poder utilizar la API de Neutron. Neutron se comunicará con el servicio de identidad para autenticar al usuario.
- **Nova**: en el proceso de creación de una máquina virtual Neutron y compute interaccionan para añadir la tarjeta de red virtual (VNIC) de la máquina virtual a una red.
- **Horizon**: en el caso de estar utilizando el servicio web para la interfaz de usuario, cada instrucción realizada, relativa a la red, se comunicará a Neutron para que este la implemente [8].

2.2.2.4 Componentes de Neutron

Los agentes expuestos con anterioridad son un conjunto de procesos que, durante su ejecución, gestionan algunas de las tareas propias de las capas de enlace y red del modelo OSI.

Se trata de funciones propias de equipos de red como switches o routers, equipos sobre los que se mapean algunos de los agentes y servicios de Neutron. Por lo que, desde este punto de vista, se tiene un conjunto de elementos utilizando distintas herramientas para implementar la API del servicio de red.

Concretamente, los componentes de cualquier red virtual implementada a través de la API de Neutron, se pueden resumir en un conjunto de switches virtuales, Linux bridges, interfaces de red y espacios de nombres.

Debido a la gran dependencia que tiene la implementación de la API con la tecnología instalada como plugin en el Neutron-L2-plugin-Agent es necesario concretar cuál se utilizará durante esta explicación.

Puesto que se introdujo en la versión Habana de Openstack con el objetivo de remplazar OpenvSwitch y Linux Bridge, pero continúa permitiendo su utilización integrándolos como drivers, se utilizará como base para esta explicación el plugin Modular Layer 2 utilizando OpenvSwitch como driver.

Interfaces de red

Entre los dispositivos de red esenciales en Neutron para la transmisión y recepción de tramas se encuentran dos tipos de interfaces [8]:

- Los dispositivos de red conocidos como Terminal Access Point (TAP) son interfaces de red virtuales implementadas en los hipervisores para simular las tradicionales Network Interface Card (NIC). Se trata de un componente esencial para cualquier hipervisor que quiera tener acceso a la red. Es el encargado de hacer llegar al sistema operativo todas las tramas Ethernet que reciba.
- Los pares veth son interfaces de red virtuales directamente conectadas entre sí. Funcionan como un cable virtual que establece una conexión entre bridges virtuales.

Linux Bridge

En este tipo de implementación se encuentran equipos Linux Bridge cumpliendo una función distinta de la habitual. Cada Linux bridge actúa como un hub al que se pueden adjuntar una o varias interfaces de red. Cada paquete recibido se retransmite por el resto de las interfaces, actuando efectivamente como un hub estándar.

El Linux bridge, en teoría, no es un equipo estrictamente necesario para la conexión entre instancias y un bridge OpenvSwitch. Es suficiente con adjuntar la interfaz TAP de cada instancia directamente a un puerto del bridge OpenvSwitch. Sin embargo, los grupos de seguridad que utiliza Openstack para implementar firewalls a nivel de instancia utilizan Iptables, una utilidad no soportada por OpenvSwitch, pero si por los Bridge de Linux [8].

La introducción de un Linux Bridge entre cada instancia y el bridge OpenvSwitch soluciona este problema. La interfaz TAP se conecta directamente al Linux Bridge, mientras que la conexión entre este y el bridge OpenvSwitch se realiza con un par de interfaces veth como se aprecia en la figura 7.

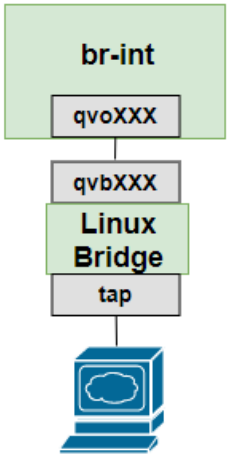

Figura 7. Estructura del Linux Bridge.

Espacios de nombres

Los espacios de nombres son una tecnología de kernel esencial para la creación de contenedores. Se trata de una abstracción de un recurso global del sistema como puede ser la pila de recursos de red de Linux. De esta manera, los procesos asociados a este recurso global del sistema se encuentran aislados dentro del espacio de nombres [14].

Gracias a los espacios de nombres se consigue el aislamiento de recursos, es decir, cada espacio de nombres dispone de una serie de recursos del sistema totalmente independientes del resto de espacios.

En este contexto un espacio de nombres de red supone un aislamiento de la pila de recursos de red. Cada espacio dispone de sus propias interfaces de red, tablas de rutas y regla de iptables.

Openstack utiliza esta tecnología para proporcionar a cada tenant un esquema de direccionamiento, tablas de rutas y otros servicios de red independientes del resto de usuarios. Concretamente, se utiliza para implementar los servicios dhcp y de capa de red con los nombres de espacios qdhcp y qrouter respectivamente.

Por cada red tenant se implementa un espacio de nombres qdhcp para ofrecer servicio dhcp a las instancias de dicha red. La interacción con el bridge de integración se realiza a través de una interfaz tap-XXX, adjuntada a uno de sus puertos. De igual manera, por cada router de un tenant, se implementa un espacio de nombres qrouter. Estos espacios se basan en las interfaces qg-XXX y qr-XXX para realizar las operaciones necesarias en cada paquete. Tanto el espacio de nombres qdhcp como el qrouter se implementan en el nodo de red e interaccionan con bridge de integración de dicho nodo.

La estructura interna del espacio de nombres qrouter y su interacción con otros componentes se describe en la figura 8.

Las interfaces qg-XXX y qr-XXX se adjuntan a sendos puertos internos del bridge de integración. Cada red privada que se adjunta al router, lo hace por medio de una interfaz qr-XXX con una dirección ip privada, pasando a actuar dicha interfaz como gateway para las instancias de la red. Cada router que se adjunta a la red pública externa, lo hace a través de una interfaz qg-XXX con una dirección ip pública. El espacio de nombres qrouter utiliza las direcciones ip asociadas a estas interfaces a la hora de realizar operaciones como el enrutamiento o el NAT. Sustituye las direcciones ip destino u origen de los paquetes por las de estas interfaces según sea necesario.

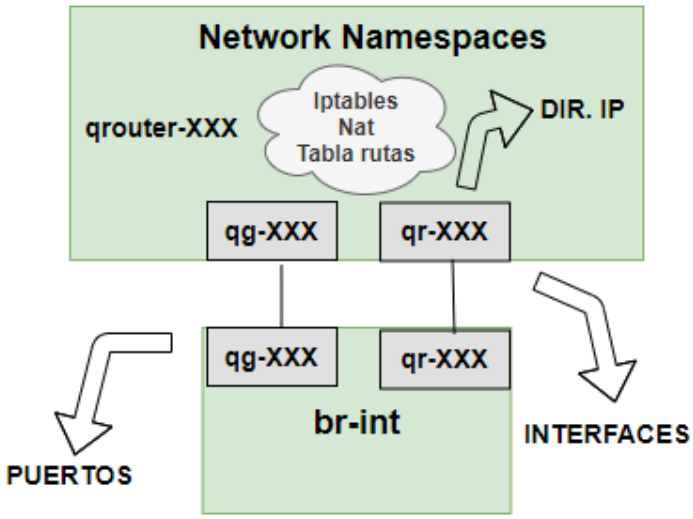

Figura 8. Estructura del espacio de nombres de red.

Bridges OpenvSwitch

Un bridge OpenvSwitch actúa como un switch virtual con puertos configurables tal y como lo serían los de un switch físico. Se distingue entre puertos externos e internos del bridge. Los primeros se caracterizan porque permiten la conexión de distintas interfaces de red virtuales como pares de interfaces veth o los Terminal Access Point (TAP). Además, son configurables mediante tecnologías como vlan, vxlan o gre. Los puertos internos no son accesibles por las herramientas nativas del sistema, solo son visibles en el contexto de OpenvSwitch, pero permiten asignar uno o varias direcciones ip en un mismo bridge.

Existen tres tipos de bridges OpenvSwitch, un bridge de integración (br-int), un bridge de conectividad física (br-ethx), llamado bridge exterior (br-ex) en el nodo de red, y un bridge túnel (br-tun) [8].

Bridge de integración (br-int)

El bridge de integración es el elemento principal en el proceso de conmutación. Se implementa en cada nodo físico que requiera el servicio de red. Existen dos posibilidades, que se localice en un nodo de cómputo o que lo haga en un nodo de red.

En el primer caso, cualquier instancia creada en el nodo se enlaza directamente al bridge. Se encarga de realizar la traducción interna de etiquetas y de conmutar las tramas. Las instancias que pertenezcan al mismo tenant serán etiquetadas con el mismo identificador en el puerto con la interfaz qvo-XXX adjunta. Por un lado, posee un puerto para cada instancia del nodo, al que permite conectar tanto interfaces TAP como veth (qvo-XXX), y por otro, posee un puerto por cada bridge con el que mantenga conexión, al que permite conectar interfaces veth.

Las tramas procedentes de instancias con destino a la red física del proveedor son enviadas con la etiqueta que identifique su red, por el puerto int-br-eth0 o patch-tun dependiendo de la tecnología utilizada, mientras que las tramas procedentes de otros bridges deben ser traducidas desde el id de túnel, en caso de que su procedencia sea el br-tun, o desde la etiqueta externa, en caso de proceder del br-ethx, a la etiqueta interna correspondiente. En el caso de las tramas procedentes de otro bridge, el bridge procede a reenviar la trama por los puertos que pertenezcan a la red identificada por dicha etiqueta interna. La figura 9 muestra la estructura interna de un bridge de integración y su interacción con otros componentes en un nodo de computo.

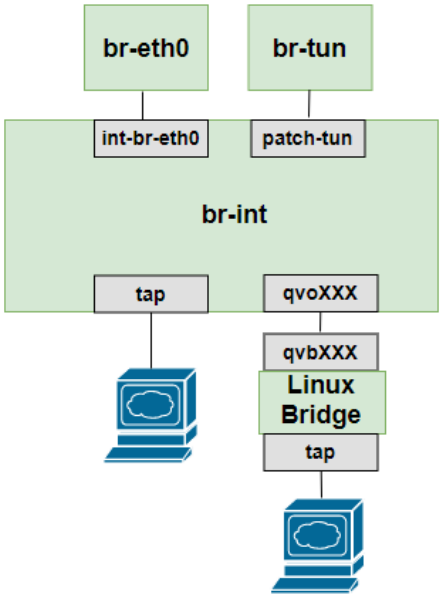

Figura 9. Estructura del bridge de integración del nodo de cómputo.

En el segundo caso, el bridge de integración presenta una estructura similar a la figura 10. Por un lado, consta de tres puertos para la interacción con otros bridges que ofrecen acceso a las redes físicas del proveedor. Esta interacción es posible gracias a pares de interfaces veth que establecen una vía de comunicación. Por otro lado, consta de otros tres tipos de puertos para interaccionar con espacios de nombres como qdhcp o qrouter. El servicio dhcp se implementa sobre la interfaz tap-XXX, mientras que el espacio de nombres qrouter lo hace por medio de las interfaces qg-XXX y qr-XXX. Estas interfaces disponen de una dirección ip propia. La interfaz qr-XXX actúa como gateway de la red privada sobre la que se mapea, mientras que la interfaz qg-XXX se mapea sobre una red pública externa que comparten todos los tenant.

En este caso, el procesamiento al que se ve sometido un paquete depende de la red física de la que proviene.

Aquellos paquetes procedentes de la red física externa atraviesan el bridge external hasta llegar a la interfaz int-br-ex en el bridge de integración. En función de la dirección ip de destino el br-int reenvía el paquete a la interfaz qg-XXX

correspondiente. Es aquí donde el espacio de nombres realiza las operaciones oportunas sobre el paquete, entregándolo a la interfaz qr-XXX correspondiente. La interfaz qr-XXX actúa como gateway de una red privada en concreto. La interfaz añade la etiqueta interna conforme a la red privada asociada. Finalmente, el bridge de integración reenvía la trama etiquetada al bridge túnel o al bridge ethx a través de la interfaz patch-tun o la interfaz int-br-ethx respectivamente.

En sentido contrario, las tramas son recibidas y procesadas por la interfaz patch-tun o la interfaz int-br-ethx. En cualquier caso, se realiza la traducción del identificador externo con el que llegan a la etiqueta interna correspondiente. En este punto, el bridge conmuta la trama hasta la interfaz qr-XXX asociada a dicha etiqueta interna. El espacio de nombres qrouter efectúa las operaciones adecuadas sobre el paquete y lo devuelve al bridge de integración. Hay varias posibilidades en función del destino del paquete.

En un destino externo al cloud, el paquete es entregado en la interfaz qg-XXX del bridge de integración y posteriormente reenviado al bridge external por la interfaz int-br-ex.

En un destino interno, el paquete es entregado en la interfaz qr-XXX conforme a la red privada del destino. La interfaz añade la etiqueta interna correspondiente. El bridge reenvía la trama etiquetada al bridge túnel o al bridge ethx para terminar.

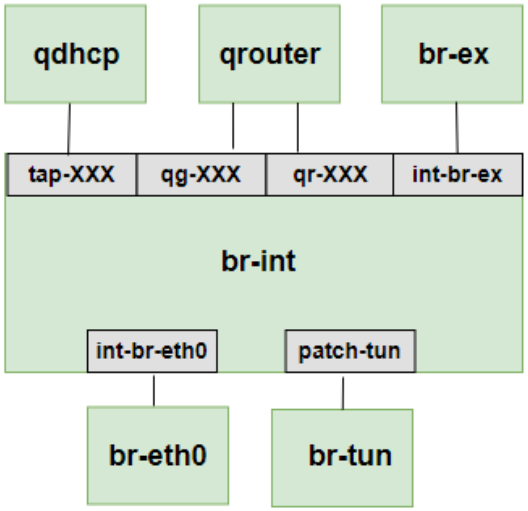

Figura 10. Estructura del bridge de integración del nodo de red.

Bridge túnel (br-tun)

El bridge túnel ofrece acceso a la red física de la infraestructura del proveedor. Hace de puente entre el bridge de integración y la red física interna del proveedor. Se utiliza cuando la creación de redes tenant ha sido previamente configurada para segmentar la red física, es decir, virtualizar la red con túneles

virtuales sobre la de capa de red. Esto sucede al utilizar tecnologías como vxlan o gre. El bridge consta de un puerto para la conexión con el bridge de integración, al que permite conectar interfaces veth (patch-int), y tantos puertos gre, conectados a la interfaz física del nodo, como túneles virtuales se establezcan.

En este caso, la estructura del bridge es independiente del nodo físico en el que se localice. Cuando recibe un paquete procedente del bridge de integración será procesado en el puerto con la interfaz patch-int. Es aquí donde se realiza la traducción entre la etiqueta interna con la que llega el paquete al ID del túnel asociado a dicha etiqueta. A continuación, el paquete es enviado por el puerto gre asociado a dicho túnel. Los paquetes entrantes desde la red física son enviados al interfaz del túnel asociada al ID de túnel del paquete. El bridge elimina las cabeceras ip y gre para añadir la etiqueta interna asociada a dicho túnel. El proceso termina enviando la trama hacia el br-int a través de la interfaz patch-int. La figura 11 muestra la estructura interna del bridge.

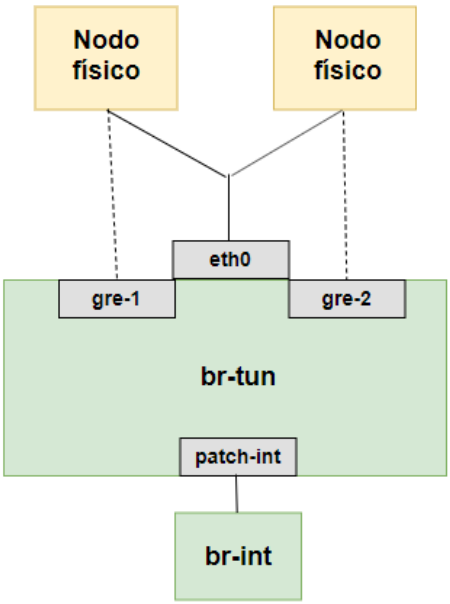

Figura 11. Estructura del bridge túnel.

Bridge conectividad física

Bridge ethx (br-ethx)

Cumpliendo una función similar a la ofrecida por el bridge túnel está el bridge ethx y al igual que él se localiza en cada nodo de la infraestructura. La principal diferencia se encuentra en la tecnología utilizada para la creación de redes virtuales. A diferencia del bridge túnel, utilizado para la creación de túneles virtuales sobre la capa de red, este bridge se utiliza cuando se segmenta la red física mediante la tecnología vlan o directamente cuando se implementan redes tipo flat, sin ningún tipo de virtualización sobre la red física. Es decir, cuando la tecnología implicada en la virtualización de la red trabaja a nivel de enlace, sin intervención de la capa de red.

Las tramas reenviadas desde el bridge de integración son recibidas y procesadas en la interfaz phy-br-eth0. La etiqueta interna es traducida a una etiqueta externa y posteriormente reenviada por la red física. Otro nodo recibe la trama a través de su interfaz física y realiza el proceso inverso, traduciendo la etiqueta externa a la etiqueta interna asociada. La trama termina reenviándose al bridge de integración a través de la interfaz phy-br-eth0. Como se ha mencionado anteriormente, todas las etiquetas, internas y externas implicadas en el proceso, identifican a la misma red privada. La figura 12 muestra la estructura interna del bridge y su interacción con otros componentes.

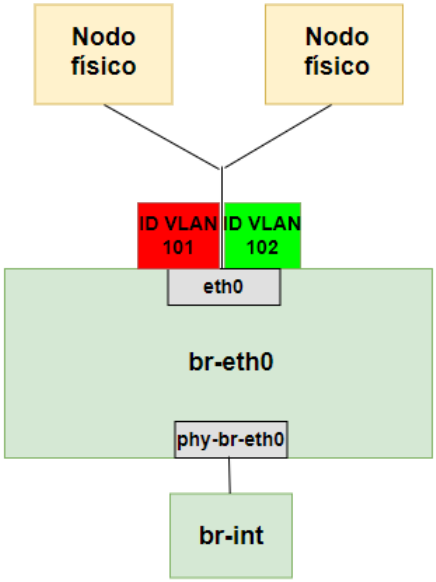

Figura 12. Estructura del bridge de conexión física.

Bridge external (br-ex)

En el nodo de red se da una situación excepcional. El nodo tiene acceso a dos redes físicas, por lo que necesita un bridge sobre el que mapear cada red. El acceso a la red física interna, encargada de conectar los nodos de computo, está cubierto por el bridge-ethx que se acaba de exponer. Sin embargo, el bridge que da acceso a la red física externa es conocido habitualmente como br-ex y posee unas características similares a cualquier br-ethx. La principal diferencia se encuentra en el tipo de segmentación utilizado para virtualizar la red. Aunque no tiene por qué ser siempre así, el br-ex suele implementar una red virtual tipo flat para mapear la red física. La figura 13 muestra la composición del bridge external.

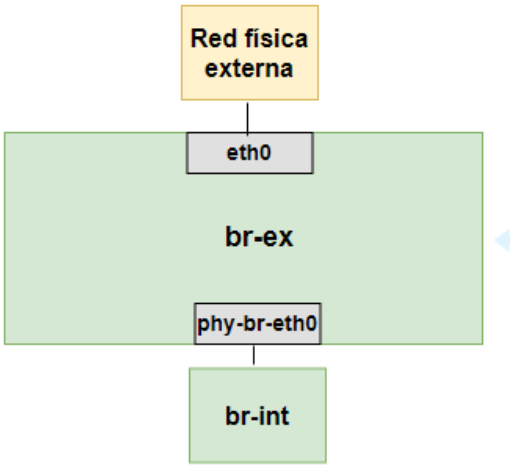

Figura 13. Estructura del bridge external.

2.2.3 Modelos de servicio

En Openstack el proveedor de servicios de nube cuenta con dos opciones a la hora de entregar su servicio de red a los clientes. Los distingue como modelo proveedor y modelo self-service. Para poder entender estos dos estilos de ofrecer el servicio de red, es importante conocer que tipos de redes se pueden crear en Openstack y que atributos son configurables en este proceso.

Tipos de red

En primer lugar, es importante diferenciar entre las redes físicas, encargadas de conectar equipos físicos como los nodos de la infraestructura, y las redes virtuales que permiten conectar las interfaces de las instancias y de otros agentes de red a sus puertos. Este tipo de redes se generan sobre los recursos físicos de la nube por lo que no es necesario ningún equipo de red adicional para implementarlas.

Dentro del grupo de redes virtuales, Openstack hace una distinción en función del creador de la red. Por un lado, si la red es creada por un administrador de la nube, será conocida como red de proveedor, mientras que, si la red es creada por un usuario de un tenant será conocida como red tenant.

Las redes de proveedor se caracterizan, aparte de por ser creadas por administradores, por ser mapeadas sobre redes físicas, permitiendo el acceso directo a recursos que no pertenecen al entorno de virtualización creado por Openstack. Para realizar este mapeo el administrador de la red debe configurar los siguientes atributos en el proceso de creación de una red [8]:

- Tipo de segmentación (flat, vlan, geneve, vxlan, gre)
- ID de segmentación (ID de vlan, ID de túnel)

- Nombre de red física

Sin embargo, las redes de proveedor no son las únicas que necesitan definir estos atributos durante su creación. Las redes tenant también utilizan estos atributos, ya que, aunque no estén disponibles para el usuario en el momento de su creación, estas se mapean sobre una red física cuando se trata de una infraestructura multinodo. Es por ello, por lo que en el proceso de creación se necesita de un mecanismo para la segmentación de la red física. El mecanismo es seleccionado directamente por Neutron, haciendo uso de la configuración del plugin, es decir, haciendo uso de la configuración previamente establecida por el administrador de la nube.

El atributo tipo de segmentación hace referencia al mecanismo físico mediante el cual se implementan las redes virtuales que se asignan directamente sobre la red física. Las redes de proveedor permiten utilizar cualquiera de los mecanismos vistos, típicamente vlan o flat, mientras que las redes tenant utilizan normalmente vlan, vxlan o gre.

El atributo id de segmentación se utiliza para asignar un identificador único a la red virtual dentro de un mismo segmento de red física. Existen dos escenarios en función del mecanismo elegido para la segmentación o virtualización de la red física. En caso de utilizar como mecanismo vlan, es necesaria una configuración del switch físico para que realice la conmutación en función del atributo id de segmentación de la red virtual y del puerto de entrada al switch. Al contrario que con vlan, los mecanismos vxlan y gre no precisan de esta configuración, utilizan encapsulamiento ip para solventar este problema.

Otro atributo relevante en la creación de redes virtuales es el nombre de la red física sobre la que se mapean. Exceptuando a las redes utilizando vxlan o gre como mecanismos para la virtualización que no necesitan definir ningún nombre.

Las redes vlan son implementadas mediante paquetes sobre la red física. Estos paquetes contendrían cabeceras del protocolo IEEE 802.1Q con un valor de campo VID específico. Las redes vlan que comparten la misma red física se encontrarían completamente aisladas unas de otras. Cada red física distinta que admite redes vlan se trata como una troncal vlan separada, con un espacio distinto de valores VID. Los valores VID válidos son del 1 al 4094. Estas redes necesitarían de una correspondencia con la configuración del switch físico [8].

Las redes gre o vxlan son muy similares entre sí, cada red de este tipo recibe un identificador único de túnel. Se trata de redes implementadas sobre tecnologías de capa de enlace y de red ya existentes, con el fin de lograr arquitecturas de red más flexibles. Como se puede ver en la figura 14, la implementación de un túnel gre se realiza mediante la encapsulación de tramas Ethernet. Se añade una cabecera gre y una cabecera ip sobre esta. De este modo, cada nodo mantiene su propia tabla de rutas a la hora de enviar los paquetes sobre la red. La principal diferencia con las redes vlan se encuentra en la falta de una configuración adicional en el switch físico [8].

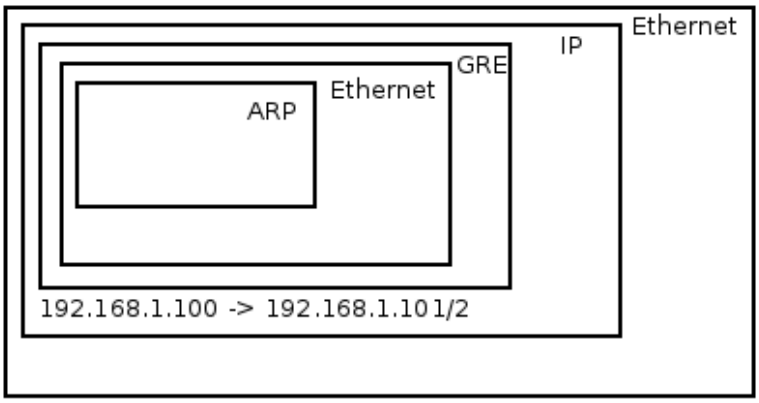

***Figura 14. Encapsulamiento GRE* [12].**

Aunque se acaba de ver que es posible, no es práctico utilizar vlan en redes tenant, mientras existan otras tecnologías con mayor escalabilidad como vxlan o gre. Es preferible utilizar vlan en redes de proveedor, quedando como principal mecanismo junto al conocido como flat. El tipo de segmentación flat se utiliza sobre redes físicas en las que no se pretende realiza ningún tipo de virtualización. Esto implica que todos los equipos en este tipo de redes son visibles entre sí.

Por último, es importante conocer los matices que aportan otros atributos como router-external y shared [15].

Combinando los posibles valores de estos atributos se obtienen múltiples tipos de red que permiten definir de forma sencilla los dos modelos que existen de ofrecer el servicio de red.

Se trata de dos atributos con un posible valor de falso o verdadero, por lo que surgen las cuatro siguientes combinaciones:

- Ambos atributos puestos en falso durante la creación de la red. Estamos ante el típico escenario de una red tenant, creada por el administrador o por el usuario, accesible a todos los miembros de un tenant. Como se ha visto antes, se utilizará VXLAN o GRE para la virtualización de la red.
- Router-external puesto como falso y shared como verdadero. Aquí existen dos posibilidades. O bien se trata de redes tenant que pueden ser compartidas mediante acceso basado en roles (RABC), o se trata de redes de proveedor, accesibles para todos los tenant, de forma que se les permite conectar instancias directamente pero no routers virtuales. Todos los tenant comparten el mismo esquema de red.
- Router-external como verdadero y shared como falso. Se trata de un tipo de red de proveedor conocido como red externa. Como red de proveedor que es, solo puede ser creada por el administrador. Cualquier tenant tiene acceso a esta red a través de un router virtual creado por el Neutron-L3-Agent. Esto implica que cada tenant tiene su propio esquema de direccionamiento. Típicamente utiliza VLAN o FLAT como mecanismos de virtualización.

- Ambos atributos puestos como verdadero. Es el típico caso de una red de proveedor que únicamente puede ser creada por el administrador. Es accesible a todos los tenant y estos pueden conectar directamente sus instancias a ella. Esto significa que se comparte el esquema de direccionamiento.

El atributo router-external hace referencia a la capacidad de un tenant para conectar la interfaz externa de su router virtual a la red. Por lo que se puede deducir que el último escenario es una ampliación del tercero.

Los matices que aportan estos dos atributos permiten definir los fundamentos sobre los que se construyen los dos modelos de red ofrecidos por Openstack.

Modelos de red

En el modelo provider red, el administrador busca que cada tenant tenga un grupo de direcciones fijas asignadas a su completa disposición. De este modo se logra que él usuario no tenga que realizar complejas topologías de red y pueda conectar sus instancias directamente a la red. El administrador crea las redes virtuales para cada tenant utilizando vlan. Durante la creación de estas redes utiliza atributos que coincidan con las configuraciones previamente establecidas en sus equipos de red físicos. Otra posibilidad en este modelo es que el administrador utilice flat como mecanismo para la segmentación de la red física.

La figura 15 muestra una nube implementada bajo un modelo provider red. El administrador configura una red de proveedor que se mapea sobre la red física utilizando vlan como mecanismo para la virtualización. De esta forma puede alojar tenants con distintas subredes. Los tenant tendrán la posibilidad de lanzar directamente las instancias sobre estas subredes. Los servicios de enrutamiento o NAT tienen lugar en el router físico. Aunque en el ejemplo se ve que cada tenant hace uso de subredes con distinto direccionamiento, se sigue compartiendo el mismo espacio de direcciones, lo que significa que no puede haber superposición de direcciones entre tenants.

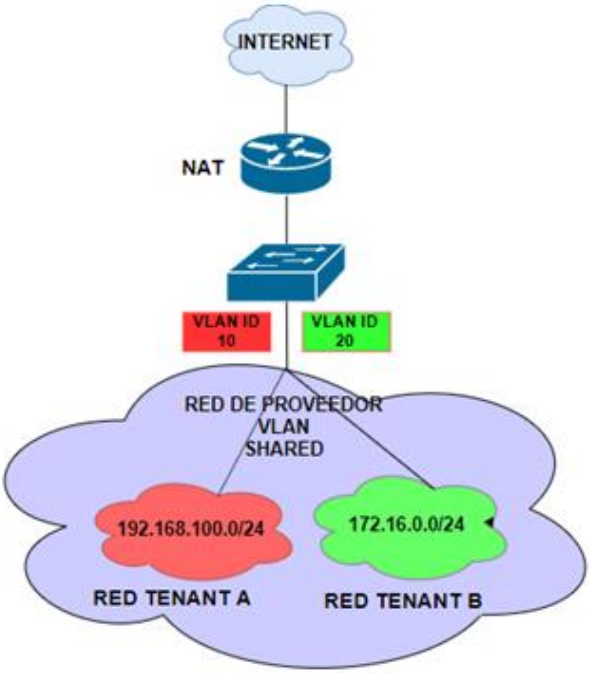

Figura 15. Nube basada en el modelo red de proveedor Vlan.

En la figura 16, se ejemplifica el segundo caso. El administrador decide implementar el cloud siguiendo un modelo provider red. Crea una red virtual de proveedor con el fin de ofrecer soporte a los clientes. El mapeo sobre la red física se efectúa mediante flat, lo que provoca que todos los tenant compartan el mismo espacio de direccionamiento y la misma subred para lanzar sus instancias, en este caso con dirección 172.24.4.0/24. AL igual que antes, todos los servicios relacionados con el nivel de red se implementan directamente sobre el router físico.

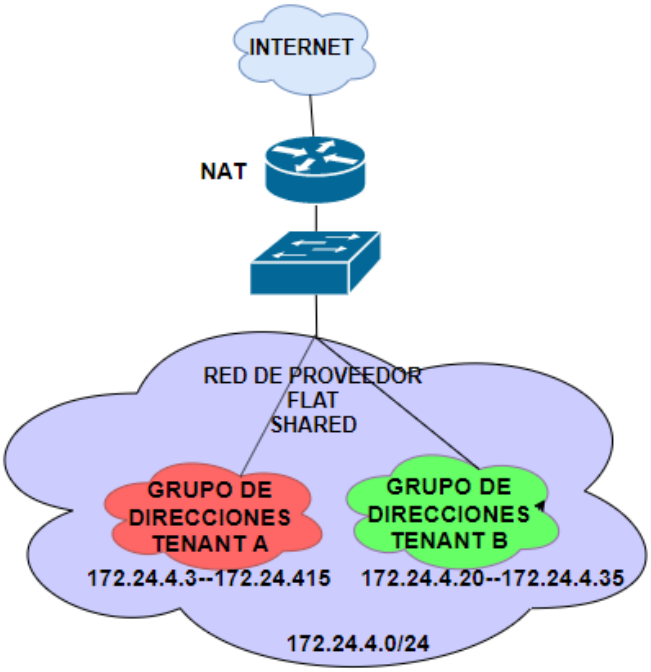

Figura 16. Nube basada en el modelo red de proveedor flat.

En el modelo self-service, el administrador de la nube busca que sus usuarios puedan crear complejas topologías de red de acuerdo con las necesidades específicas de cada uno. Para ello, implementa una red virtual de proveedor que mapea sobre la red física utilizando como mecanismo flat. El administrador debe llevar a cabo una serie de configuraciones sobre sus equipos físicos, así como sobre el plugin instalado en cada nodo de la infraestructura. Debe realizar estas configuraciones con el fin de que el usuario pueda implementar sus redes virtuales sin tener conocimiento del proceso físico mediante el cual se lleva a cabo.

Para tener acceso al exterior de la nube, el administrador permite que cada tenant conecte la interfaz pública de sus routers a la red de proveedor. Cada router conectado a la red dispondrá de una dirección ip dentro de la red de proveedor. Los router compartirán esquema de direccionamiento y subred, de modo que no habrá superposición de direcciones entre routers. Además, cada usuario dispondrá de un grupo de direcciones, conocidas como ip flotantes, pertenecientes a la red de proveedor con el objetivo de que cada tenant pueda mapear sus instancias sobre esta red.

El modelo self-service permite que cada tenant cree múltiples subredes virtuales mediante mecanismos como vxlan o gre. La elección de este mecanismo no compete al usuario. El usuario simplemente crea la red y Neutron, a través del plugin previamente configurado, ejecuta la virtualización. Además, se consiguen esquemas de direccionamiento independientes entre tenants, por lo que podrá haber superposición de direcciones.

Como se muestra en la figura 17 cada tenant conecta su router a la red de proveedor con dirección 172.24.4.0/24. Cada router consigue una dirección única dentro de esta red. Sin embargo, cada tenant dispone de un esquema de direcciones completo para su total manipulación.

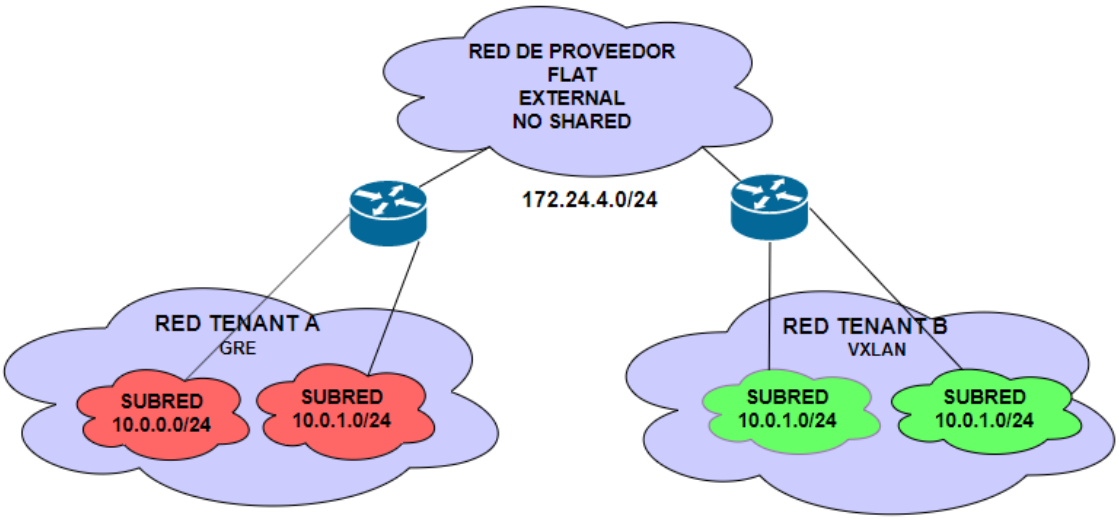

Figura 17. Nube basada en el modelo self-service.

3 Escenarios de vulnerabilidad

El proveedor de servicios de nube se enfrenta a distintas situaciones en las que la seguridad de los recursos contratados por algún cliente se puede ver comprometida. En el caso particular de las nubes privadas, la organización propietaria de la infraestructura de nube afronta situaciones en las que debe velar por la seguridad de sus recursos y por su propia reputación. Si bien es fácil suponer en que situaciones pueden quedar expuestos los recursos de una nube pública, es más complicado en el caso de una nube privada, puesto que, en principio puede parecer que al ser tan solo accesible para los propios empleados de la empresa no habrá intenciones maliciosas por parte de estos. Situaciones en las que una empresa cuenta con diversos departamentos alojados en la misma nube requieren de ser examinados como posibles escenarios de intrusión.

Con el fin de lidiar con esta problemática, es habitual hacer uso de herramientas para la monitorización y manipulación de aquel tráfico que se considere como malicioso. Aunque existen herramientas y servicios para realizar filtrado de tráfico, como los sistemas firewall, es necesario permitir algún tipo de tráfico por mínimo que este sea, de lo contrario, el recurso queda completamente aislado sin posibilidad de ser accesible. Ante esta situación, obtienen una especial relevancia los sistemas para la detección de intrusos en red como Snort.

La combinación de estos sistemas junto con grupos de seguridad o el firewall as a service de Openstack permite detectar actividades maliciosas para su posterior filtrado, evitando que alcancen cualquier recurso de la nube. Sin embargo, es necesario decidir el emplazamiento de estas herramientas dentro de la infraestructura de la nube.

Un administrador encargado de la seguridad en una empresa debe conocer la ruta trazada por cualquier tráfico entrante y saliente de sus recursos, de forma que, pueda localizar su sistema de detección de intrusos allí donde considere oportuno.

Por esto, se dedica este apartado a exponer los escenarios de comunicación en los que quedan expuestos a posibles intrusos los recursos de un determinado tenant o departamento de una empresa.

En función del origen del tráfico se distinguen dos posibles escenarios. El origen puede situarse en el exterior de la nube e intentar acceder a través de internet, o puede estar situado en el interior de la nube, es decir, puede tratarse de un empleado de la propia empresa en otro departamento o un intruso que ha conseguido establecerse en el interior de la nube. En este caso se producen variantes dependiendo del origen del empleado. Existen tres posibilidades: situado en otra red tenant, en otra subred del mismo tenant o en la misma subred del objetivo.

Descripción de la nube privada

En ambos casos, es necesario definir previamente aspectos relativos a la implementación de la nube llevada a cabo por el proveedor o empresa en este caso, como el modelo de servicio ofrecido, el plugin del Neutron-L2-Agent elegido, la

infraestructura sobre la que se despliega la red o los tipos de segmentación utilizados en la implementación de cada red virtual. Además, se realiza la explicación con datos específicos para facilitar su seguimiento.

El análisis se lleva a cabo en una nube implementada bajo un modelo self-service en el que se permite al usuario definir la topología de su red mediante la creación de redes tenant y routers virtuales libremente.

La empresa ofrece una red de proveedor externa, compartida por todos los departamentos o tenants con el fin de dar acceso externo a estas redes privadas.

El tipo de segmentación configurado para la virtualización de las redes tenant sobre la red física es la tecnología generic routing encapsulation (GRE), basada en túneles virtuales sobre la capa de red. Para virtualizar la red de proveedor se utiliza un tipo de segmentación flat.

La infraestructura de la nube se compone de un nodo de red y un nodo de computo sobre el que se implementan las instancias de cualquier cliente.

3.1 Escenario con origen externo

En este escenario se describe la ruta trazada por tráfico procedente de internet dirigido hacia una instancia situada en la red privada de un tenant. La instancia se identifica con una dirección ip privada y otra dirección ip flotante en la red de proveedor externa.

3.1.1 Topología del escenario

Direcciones correspondientes a la topología de la red de la figura 18:

- Red de proveedor: 172.24.4.0/24.
- Subred tenant: 10.0.0.0/24.
- Pública del router del tenant: 172.24.4.10.
- Flotante y privada de la instancia: 172.24.4.101 y 10.0.0.10.

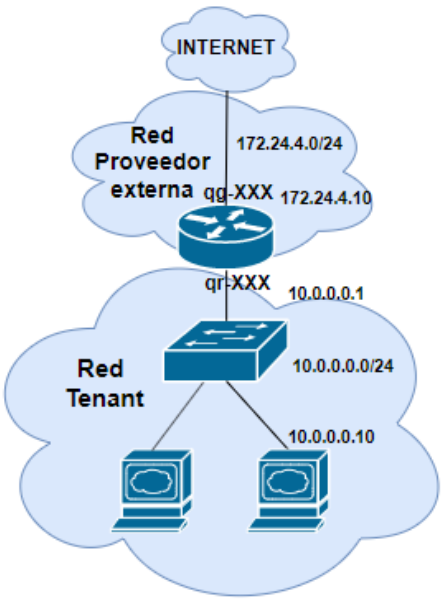

Figura 18. Topología de red del escenario con origen externo.

3.1.2 Camino de datos

Se describen los pasos en los que se ve envuelto un paquete enviado desde internet con destino a la instancia situada en la red tenant:

1. El paquete llega a la interfaz br-ex del nodo de red y es reenviado al bridge OpenvSwitch br-ex. Contiene la dirección ip flotante de la instancia, como dirección destino.
2. El bridge br-ex reenvía el paquete al bridge br-int a través de la interfaz phy-br-int.
3. Cuando el paquete se encuentra en el bridge de integración se reenvía a la interfaz qg-XXX que contenga la dirección ip flotante.
4. Una vez en la interfaz, el paquete se encuentra dentro del espacio de nombres qrouter del tenant objetivo. En este espacio de nombres se lleva a cabo el servicio DNAT, en el que se modifica la dirección ip destino del paquete. Se remplaza la dirección Ip flotante por la dirección ip privada de la instancia y se reenvía de nuevo al bridge de integración.
5. El bridge de integración recibe el paquete en el puerto con la interfaz qr-XXX adjunta, lo encapsula en la trama Ethernet con la etiqueta interna asociada a la red tenant y lo reenvía al bridge túnel a través de la interfaz patch-tun.
6. El bridge túnel añade dos nuevas cabeceras sobre la trama. En la primera añade el identificador de túnel asociado a la red del tenant y en la segunda añade la cabecera ip con la dirección destino del nodo físico al que se dirige el paquete, en este caso, con dirección ip destino correspondiente al nodo de computo. Para la segunda encapsulación se hace uso de la interfaz gre-1 asociada a este túnel físico.
7. El paquete se reenvía sobre la red física, encapsulado en una trama Ethernet con la dirección MAC destino del nodo de computo.

8. El switch físico reenvía el paquete en base a su tabla de direcciones MAC.
9. La interfaz física del nodo de computo reenvía la trama al bridge túnel. Este suprime la cabecera ip al comprobar que la dirección ip destino es su propia dirección y reemplaza el identificador de túnel por la etiqueta interna asociada a la red tenant.
10. La trama es recibida en el bridge de integración tras ser reenviada a través de la interfaz patch-int del bridge túnel.
11. La trama contiene la dirección MAC destino de la instancia, además de la etiqueta interna asociada a la red privada del tenant. Con estos detalles, el bridge de integración es capaz de reenviar el paquete por el puerto con la interfaz qvo-XXX de la instancia destino.

3.2 Escenario con origen interno

En función del origen interno surgen tres escenarios clasificados de la siguiente manera: empleado o intruso situado en otra red tenant, en otra subred del mismo tenant o en la misma subred del objetivo.

3.2.1 Distinta red tenant

En este escenario se describe la ruta trazada por tráfico generado en el interior de una red tenant de un cliente del proveedor, dirigido hacia una instancia situada en la red privada de otro tenant. Las instancias se identifican con una dirección ip privada y otra dirección ip flotante en la red de proveedor externa.

Topología del escenario

Direcciones correspondientes a la topología de la red de la figura 19:

- Red de proveedor: 172.24.4.0/24.
- Subred tenant víctima y atacante: 10.0.0.0/24 y 10.0.1.0/24 .
- Públicas del router víctima y atacante: 172.24.4.10 y 172.24.4.20.
- Flotante y privada de la instancia víctima: 172.24.4.101 y 10.0.0.10.
- Flotante y privada de la instancia atacante: 172.24.4.102 y 10.0.1.10.

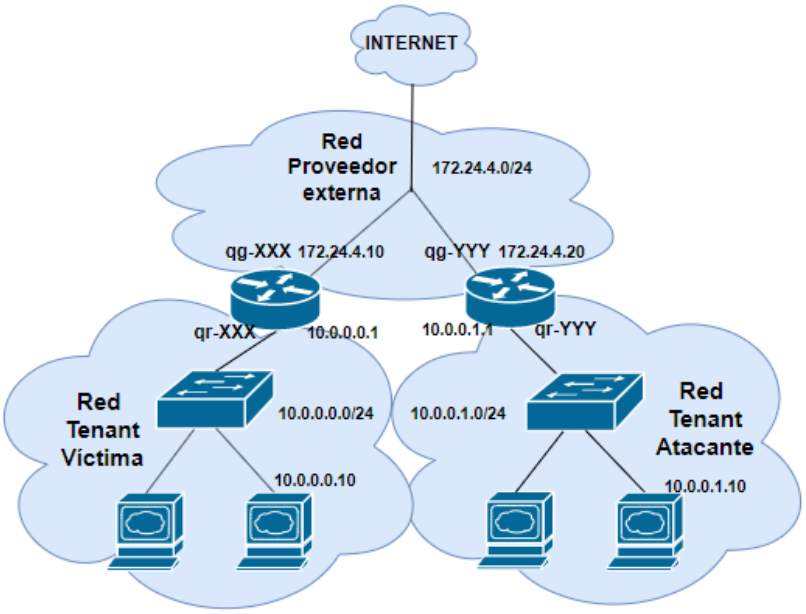

Figura 19. Topología de red del escenario con distinta red tenant.

Camino de datos

El proceso descrito a continuación envuelve cada uno de los saltos a los que se somete un paquete originado en la instancia de la red atacante con destino la instancia de la red víctima. Se distinguen tres fases en el proceso, la primera se desarrolla en nodo de computo, la segunda en el nodo de red y la tercera de nuevo en el nodo de computo. La figura 20 muestra un esquema de los componentes de red que se atraviesan durante el proceso.

- Nodo de computo:
  1. La trama se genera en la instancia y es reenviada por la interfaz tap-YYY hasta el Linux Bridge qbr-YYY. El Linux Bridge aplica el grupo de seguridad correspondiente y la reenvía a través de la interfaz qvb-YYY hacia el bridge de integración que lo termina recibiendo en la interfaz qvo-YYY.
  2. El bridge de integración añade la etiqueta interna asociada a la red tenant atacante antes de conmutarla hacia la interfaz patch-tun. La trama posee como dirección MAC destino la dirección MAC del gateway de la red, es decir, la dirección de la interfaz qr-YYY.
  3. El bridge túnel recibe el paquete y añade las cabeceras gre e ip en base a los datos de la interfaz gre-1. La cabecera gre contiene el identificador de túnel asociado a la red atacante, mientras que, la cabecera ip contiene la dirección ip del nodo de red como destino. El paquete se encapsula sobre una trama Ethernet con la dirección MAC destino del nodo de red.

- Nodo de red:
  - El nodo de red recibe la trama en su interfaz física y la reenvía al bridge túnel para ser procesada.

- El bridge túnel recibe el paquete en la interfaz gre-1, elimina la cabecera ip al comprobar que la dirección ip destino es su propia dirección y elimina la cabecera gre, reemplazando el identificador de túnel por la etiqueta interna correspondiente a la red atacante.
- El bridge de integración recibe el paquete en la interfaz patch-tun y lo conmuta, con ayuda de la etiqueta interna y la dirección MAC destino, a la interfaz qr-YYY situada en el espacio de nombres qrouter-YYY.
- El espacio de nombres qrouter-YYY realiza la operación SNAT sobre el paquete. La dirección ip origen, 10.0.1.10, se intercambia por la dirección Ip flotante, 172.24.4.102, de la instancia atacante. El paquete es devuelto a la interfaz qg-YYY del bridge de integración.
- La interfaz qg-YYY lanza una petición ARP para la dirección Ip flotante, 172.24.4.101, de la instancia víctima. El bridge de integración envía la petición por todos sus puertos, de modo que responde la interfaz qg-XXX.
- La interfaz qg-YYY reenvía el paquete con dirección ip origen, 172.24.4.102, dirección ip destino, 172.24.4.101.
- El espacio de nombres qrouter-XXX recibe el paquete procedente de la interfaz qg-XXX, realiza el servicio DNAT, intercambiando la dirección Ip flotante, 172.24.4.101, por la dirección ip privada, 10.0.0.10, de la instancia víctima, y reenvía el paquete a la interfaz qr-XXX del bridge de integración.
- El bridge de integración recibe el paquete y lo encapsula en una trama con la etiqueta interna asociada a la red víctima. Lo termina reenviando al bridge túnel a través de la interfaz patch-tun.
- El bridge túnel añade dos nuevas cabeceras sobre la trama. En la primera añade el identificador de túnel asociado a la red víctima y en la segunda añade la cabecera ip con la dirección destino del nodo de cómputo.

- Nodo de cómputo:
  1. El nodo recibe la trama en su interfaz física y la reenvía al bridge túnel para que este la procese.
  2. El bridge túnel recibe el paquete en la interfaz gre-1, elimina la cabecera ip al comprobar que la dirección ip destino es su propia dirección y elimina la cabecera gre, reemplazando el identificador de túnel por la etiqueta interna correspondiente a la red víctima.
  3. La trama es recibida en el bridge de integración tras ser reenviada por el bridge túnel.
  4. La trama contiene la dirección MAC destino de la instancia, además de la etiqueta interna asociada a la red privada del tenant víctima. Con estos detalles, el bridge de integración es capaz de reenviar el paquete por el puerto con la interfaz qvo-XXX correcta.

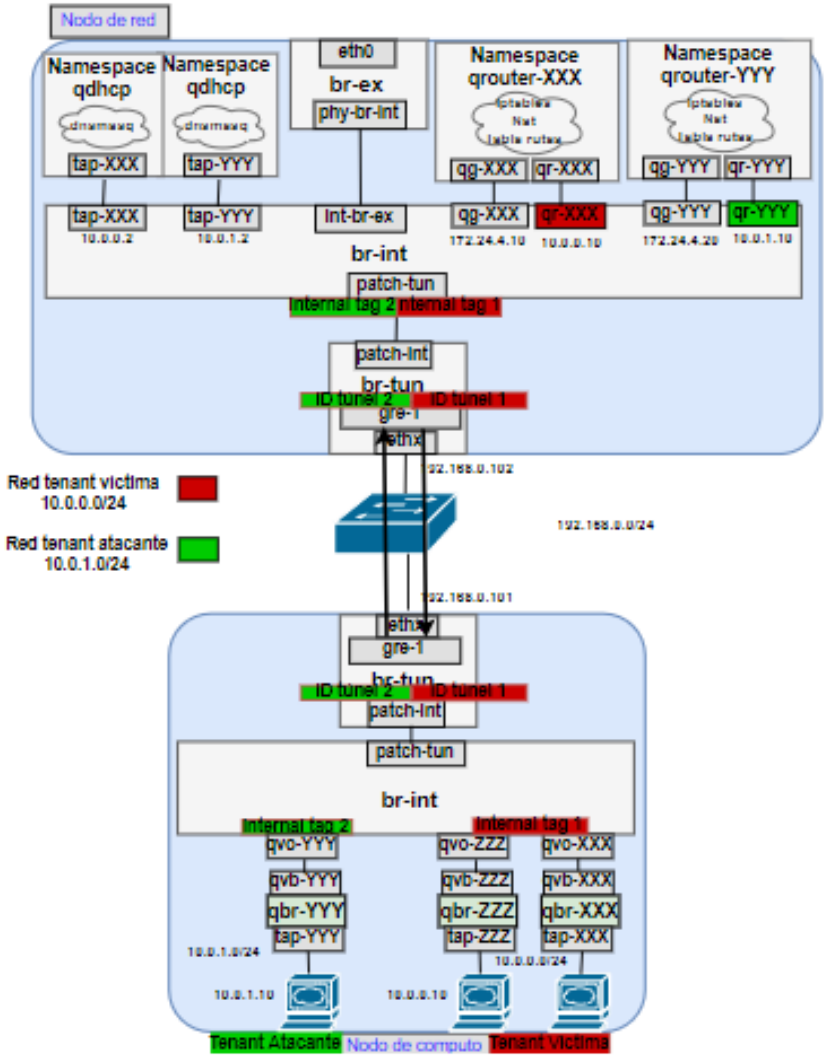

Figura 20. Estructura interna de la infraestructura con dos redes tenant.

3.2.2 Distinta subred en la misma red tenant

En este escenario se describe la ruta trazada por tráfico generado en el interior de una subred de un tenant, dirigido hacia una instancia situada en otra subred del mismo tenant. Las instancias se identifican con una dirección ip privada y otra dirección ip flotante en la red de proveedor externa.

Topología del escenario

Direcciones correspondientes a la topología de la red de la figura 21:

- Red de proveedor: 172.24.4.0/24.
- Subred víctima y atacante: 10.0.0.0/24 y 10.0.1.0/24 .
- Pública del router: 172.24.4.10.
- Flotante y privada de la instancia víctima: 172.24.4.101 y 10.0.0.10.
- Flotante y privada de la instancia atacante: 172.24.4.102 y 10.0.1.10.

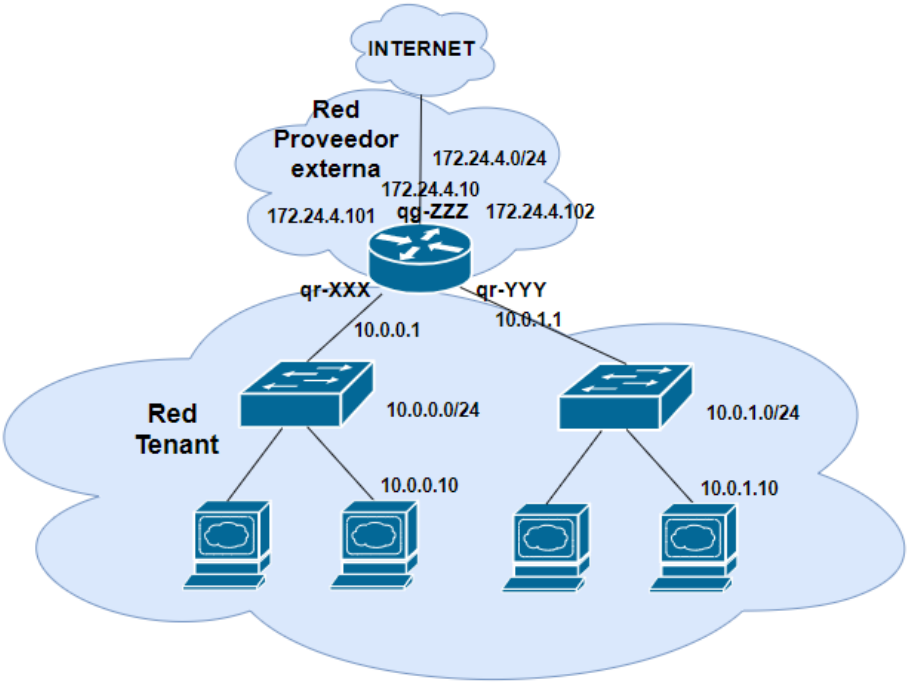

Figura 21. Topología de red del escenario con dos subredes del mismo tenant.

Camino de datos

Se describe el proceso que tiene lugar en el interior de la red tenant cuando se pretende interaccionar con la subred 10.0.0.0/24 desde la subred 10.0.1.0/24. La primera y tercera fase son similares a las descritas en el escenario con comunicación entre distintas redes tenant, la principal diferencia se encuentra en el nodo de red. La figura 22 muestra el esquema con todos los componentes implicados en el proceso.

- Nodo de red:
  1. El nodo de red recibe la trama en su interfaz física y la reenvía al bridge túnel para ser procesada.
  2. El bridge túnel recibe el paquete en la interfaz gre-1. Elimina la cabecera ip, al comprobar que la dirección ip destino es su propia dirección, y la cabecera gre, reemplazando el identificador de túnel por la etiqueta interna correspondiente a la subred 10.0.1.0/24.
  3. El bridge de integración recibe el paquete en la interfaz patch-tun y lo conmuta, con ayuda de la etiqueta interna y la dirección MAC destino, a la interfaz qr-YYY situada en el espacio de nombres qrouter-ZZZ.
  4. El espacio de nombres qrouter-ZZZ realiza la operación SNAT sobre el paquete. La dirección ip origen, 10.0.1.10, se intercambia por la dirección ip flotante, 172.24.4.102, de la instancia atacante. El paquete es devuelto a la interfaz qg-ZZZ del bridge de integración.
  5. La interfaz qg-ZZZ posee la dirección ip flotante, 172.24.4.101, de la instancia víctima, por lo que, reenvía el paquete, con dirección ip origen 172.24.4.102 y dirección ip destino 172.24.4.101, al espacio de nombres qr-ZZZ.

6. El espacio de nombres qrouter-ZZZ recibe el paquete procedente de la interfaz qg-ZZZ, realiza el servicio DNAT, intercambiando la dirección Ip flotante, 172.24.4.101, por la dirección ip privada, 10.0.0.10, de la instancia víctima, y reenvía el paquete a la interfaz qr-XXX del bridge de integración.
7. El bridge de integración recibe el paquete y lo encapsula en una trama con la etiqueta interna asociada a la red 10.0.0.0/24. Lo termina reenviando al bridge túnel a través de la interfaz patch-tun.
8. El bridge túnel añade dos nuevas cabeceras sobre la trama. En la primera añade el identificador de túnel asociado a la subred 10.0.0.0/24 y en la segunda añade la cabecera ip con la dirección destino del nodo de cómputo.

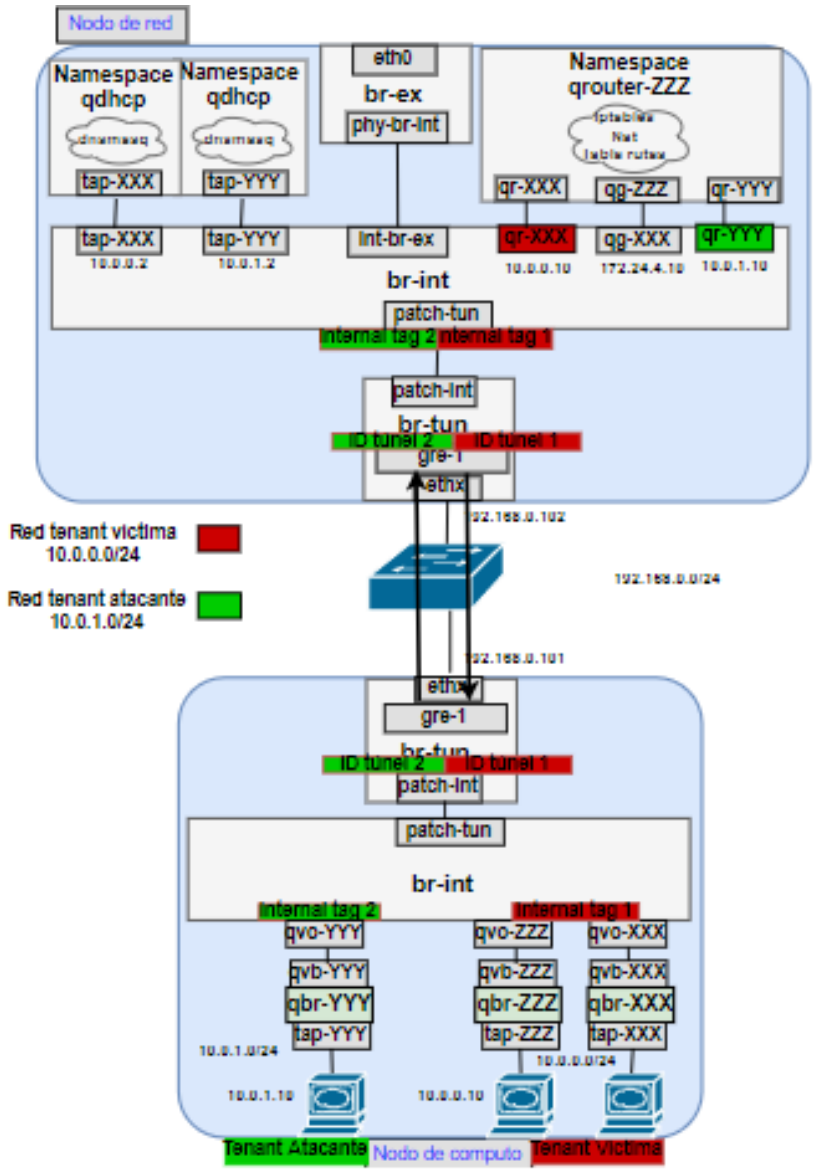

Figura 22. Estructura interna de la infraestructura con dos subredes de un tenant

3.2.3 Misma subred de una red Tenant

En este escenario se describe la ruta trazada por tráfico generado en el interior de una subred de un tenant, dirigido hacia una instancia situada en la misma subred. Las instancias se identifican con una dirección ip privada y otra dirección ip flotante en la red de proveedor externa.

Topología del escenario

Direcciones correspondientes a la topología de la red de la figura 23:

- Red de proveedor: 172.24.4.0/24.
- Subred tenant: 10.0.0.0/24.
- Dirección privada de la instancia atacante: 10.0.0.10.
- Dirección privada de la instancia víctima: 10.0.011.

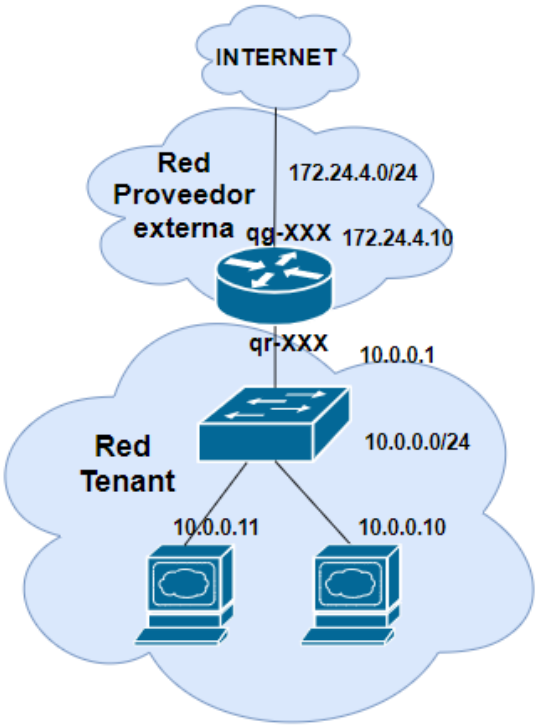

Figura 23. Topología de red del escenario con la misma subred de un tenant.

Camino de datos

Se describe el proceso que tiene lugar en el interior de la subred tenant cuando se pretende interaccionar con la instancia 10.0.0.10 desde la subred 10.0.0.11. La figura 24 muestra el esquema con los componentes implicados en el proceso.

- Nodo de computo:
    1. La trama se genera en la instancia B y es reenviada por la interfaz tap-YYY hasta el Linux Bridge qbr-YYY. El Linux Bridge aplica el grupo de seguridad correspondiente y la reenvía a través de la interfaz qvb-YYY hacia el bridge de integración que lo termina recibiendo en la interfaz qvo-YYY.

2. El bridge de integración añade la etiqueta interna asociada a la subred tenant antes de conmutarla.
3. La trama contiene la dirección MAC destino de la instancia A, además de la etiqueta interna asociada a la subred tenant. Con estos detalles, el bridge de integración es capaz de reenviar el paquete por el puerto con la interfaz qvo-XXX correcta.

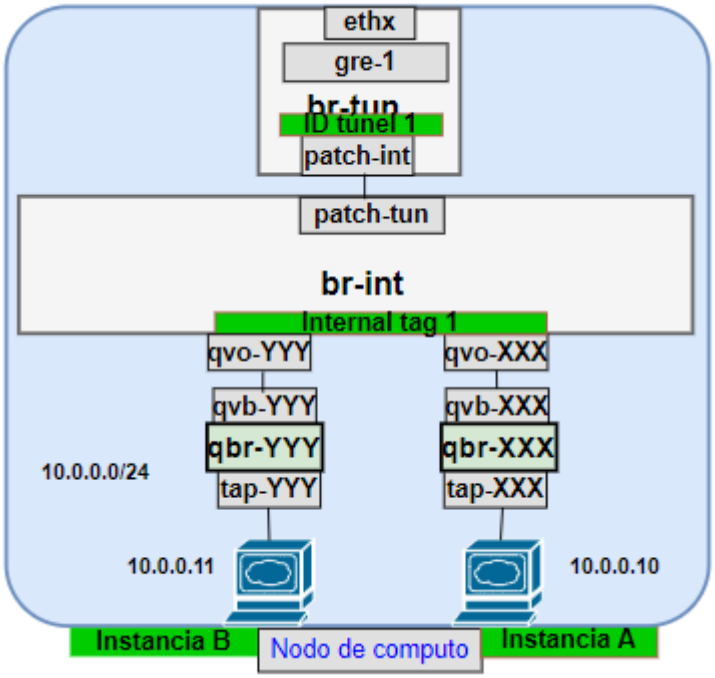

Figura 24. Estructura interna de la infraestructura con una subred.

4 Localizaciones de Snort

En una nube pública el cliente confía la seguridad de sus datos al proveedor de servicios de la nube. Sin embargo, el cliente posee servicios y herramientas suficientes para implementar una arquitectura de seguridad acorde a sus exigencias. Entre estos servicios, no se encuentra ninguno dedicado a la detección de intrusos debido a la alta complejidad que esto supondría. Por ello, se describe una posible solución a la falta de este servicio.

En una nube privada la seguridad de los recursos y de los datos almacenados en ella es responsabilidad directa del propietario de la infraestructura, que puede contratar a un tercero especializado en estos servicios o implementarlos el mismo. En este último caso, se ha de realizar un análisis repasando las posibles configuraciones y sus localizaciones antes de proceder con cualquier implementación.

4.1 Solución nube pública

Los usuarios de Openstack tienen la posibilidad de implementar su propio servicio de seguridad combinando la funcionalidad ofrecida por un sistema de detección de intrusos y un firewall. Sin embargo, esta implementación cuenta con varios puntos débiles. Se trata de una arquitectura de seguridad formada por una máquina virtual con un software de detección de intrusos instalado y un grupo de seguridad actuando como firewall en los puertos de la red privada.

La implentación tiene lugar en un nodo de computo. El usuario posee un determinado número de instancias conectadas al bridge de integración. Su objetivo es monitorizar el tráfico entrante y saliente de todas estas instancias, mediante un NIDS que alerte al administrador del tenant en caso de detectarse actividades maliciosas. El NIDS se sitúa en otra instancia conectada al mismo bridge de integración.

Es necesario duplicar el tráfico entrante en los puertos del bridge de integración dedicados a la conexión de instancias. Los equipos de red encargados de la conmutación de paquetes cubren esta necesidad mediante la técnica conocida como port mirroring, en inglés, o puertos espejo, en español.

Aunque esta funcionalidad está implementada en los bridges OpenvSwitch, un usuario de la nube no tiene acceso directo al nodo de computo donde están ejecutándose sus instancias, por tanto, no tiene acceso directo al bridge de integración. De lo contrario, esto supondría un evidente problema de seguridad, existiendo la posibilidad de manipular puertos del bridge con instancias de otros propietarios conectadas.

Esta situación implica que debe existir un servicio, en este caso ofrecido por los desarrolladores de Openstack, que permita al administrador de un tenant llevar acabo el port mirroring en aquellos puertos del bridge de integración que pertenezcan a su red privada virtual.

El servicio tap as a service se desarrolló con este objetivo. Se trata de una extensión del servicio Neutron cuya API utiliza dos objetos para su implementación.

El objeto TapService representa el puerto donde el tráfico reflejado se entrega, mientras que, el objeto TapFlow representa el puerto desde el cual el tráfico necesita ser reflejado. El principal problema del servicio se halla al no contar con soporte para puertos del servicio dhcp, ni para ningún router virtual, ni por tanto, para direcciones ip flotantes.

Con ambos objetos, un administrador de un tenant puede conectar su sistema de detección de intrusos al objeto TapService y conectar una o varias de las instancias que deben ser monitorizadas al objeto TapFlow, consiguiendo de este modo, recibir en la máquina virtual con el NIDS instalado, todo el tráfico destinado a sus instancias. Con esto, el administrador recibe alertas al detectar posibles ataques y actúa configurando grupos de seguridad acordes a las exigencias del ataque. La implementación de esta solución se describe en la figura 25.

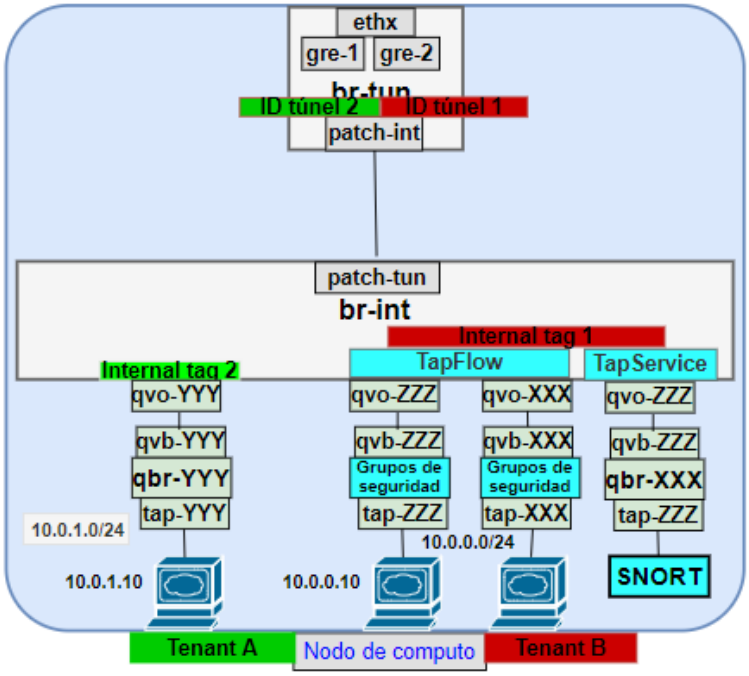

Figura 25. Solución nube pública.

4.2 Soluciones nube privada

El administrador de la nube tiene acceso a los recursos físicos de la implementación, de forma que surgen nuevas posibles localizaciones para el sistema de detección de intrusos.

La elección de estas localizaciones debe realizarse en base a factores como la cantidad de tipos de tráfico que se analiza o la cantidad de sistemas necesarios en relación con la cantidad de tráfico cubierto por cada uno.

En una nube privada con una infraestructura multinodo se sugieren tres nuevas localizaciones para monitorizar el tráfico en busca de actividades maliciosas.

4.2.1 Sistema de detección de intrusos en el nodo de red

La primera configuración consta de un sistema de detección de intrusos situado en el nodo de red, es decir, instalado directamente en el nodo, escuchando tráfico en alguna de sus interfaces. Es aquí donde se vislumbran dos posibilidades a la hora de poner a escuchar el NIDS. La figura 26 muestra esta nueva localización con Snort conectado en cada una de las posibles interfaces.

La primera posibilidad pasa por situar el sistema de detección de intrusos escuchando en el bridge external. En esta situación los factores clave en la elección se encuentra en el siguiente punto:

El tráfico capturado corresponde a tráfico entrante o saliente de la nube, con lo que se consigue protección ante intrusos externos y se mantiene la reputación de la organización propietaria de la nube. La posibilidad de que existan empleados deshonestos continúa estando presente, puesto que el tráfico entre tenants o departamentos no atraviesa el bridge external.

La cantidad de sistemas necesarios para analizar este tipo de tráfico es mínima, ya que, únicamente se sitúa un NIDS en el bride external.

El impacto sobre la actuación global de la nube será elevado. Se necesita gran parte de los recursos del nodo para el análisis del tráfico. La cantidad de tráfico analizada es muy elevada al tratarse de un único punto de salida y entrada en la nube. Esto requiere de un dimensionamiento del nodo adecuado por parte del administrador.

La segunda posibilidad pasa por situar el sistema de detección de intrusos en el bridge de integración sobre alguno de sus puertos internos. La opción más lógica parece situarlo sobre la interfaz pública de un router virtual. Con esta opción se obtiene las siguientes características:

En este caso el tráfico capturado corresponde al tráfico total entrante a la nube y al tráfico parcial saliente correspondiente al tenant propietario del router. Además, se captura tráfico interno procedente de otro tenant o departamento, o de otra subred del propio tenant.

La cantidad de sistemas aumenta considerablemente, ya que, para proporcionar protección a cada departamento es necesario situar un NIDS por cada router virtual.

Se incrementa notablemente el perjuicio sobre la actuación global de la nube. Un único nodo con recursos suficientes para todos los NIDS operando en él.

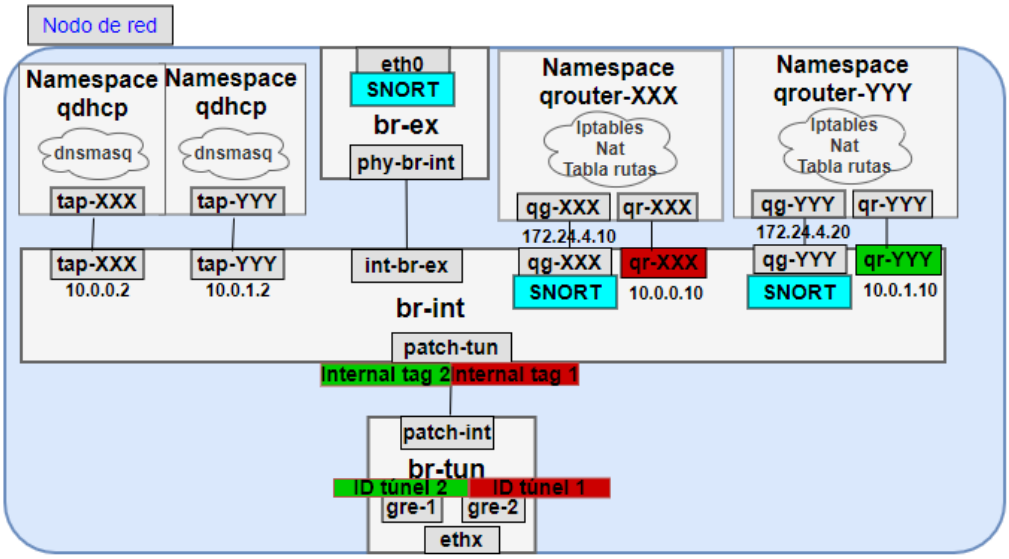

Figura 26. Solución nube privada en el nodo de red.

Es importante destacar que se utiliza como sistema de detección de intrusos, Snort. Un software que no soporta múltiples threads, por lo que La posibilidad de ejecutar Snort varias veces en el mismo nodo no existe. Esto impide la monitorización de múltiples routers al mismo tiempo.

4.2.2 Sistema de detección de intrusos en un nodo dedicado

La segunda configuración consta de un sistema de detección de intrusos situado en un nodo físico dedicado únicamente al análisis del tráfico. Se conecta directamente a un puerto del switch físico. Esta configuración implica poseer un switch físico con posibilidad de realizar port mirroring. Los factores implicados en la elección se encuentran de la siguiente manera:

En cuanto a los tipos de tráfico analizados se trata de una configuración muy completa. Es posible capturar tráfico entrante y saliente de la nube, así como, tráfico interno de otro tenant u otra subred del propio tenant. Además, la principal novedad está en que permite capturar tráfico de un mismo tenant con instancias repartidas en distintos nodos físicos.

Al tratarse de un volumen tráfico elevado es posible que se requiriese de varios nodos físicos con su sistema de detección instalado. Esto depende del tamaño de la nube.

El impacto sobre la actuación de la nube es mínimo, ya que se dedican nodos por completo al análisis. El dimensionamiento de estos nodos es responsabilidad del administrador. La figura 27 presenta el esquema de esta solución.

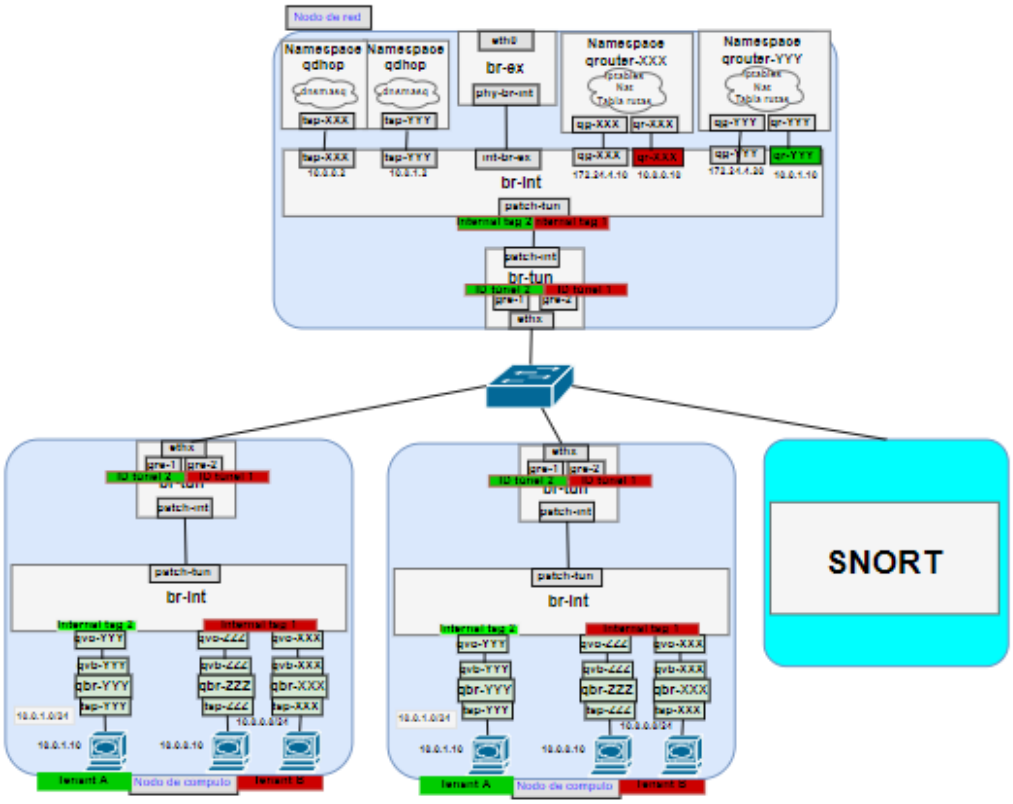

Figura 27. Solución nube privada en un nodo dedicado.

4.2.3 Sistema de detección de intrusos en un nodo de computo

La tercera y última configuración consta de un sistema de detección de intrusos instalado en cada nodo de computo. Se sitúa escuchando en la interfaz qvo del bridge de integración. Las características de esta configuración se describen a continuación:

El rango de tipos de tráfico capturado se amplía, puesto que, se monitoriza todo el tráfico de una instancia. El tráfico externo tanto entrante como saliente es analizado, el tráfico interno procedente de otro tenant u otra subred del mismo tenant también, la principal diferencia se encuentra en el tráfico de la misma red. Este tipo de tráfico depende de si la red existe únicamente en un nodo físico o si lo hace a través de varios nodos. El segundo caso se describe en la segunda configuración, mientras que, el primer caso solo es analizado en esta configuración. Este tipo de tráfico no sale del nodo de computo. Se gestiona por completo en el bridge de integración.

El número de sistemas de detección requeridos aumenta de una manera importante. Se necesita un sistema para cada instancia de la nube, lo que la convierte en una configuración adecuada tan solo en casos muy concretos donde se desee proteger un recurso específico.

La actuación global de la nube no se ve tan perjudicada como en el nodo de red, tan solo los nodos donde se realice la monitorización. Sin embargo, del mismo modo que en la primera configuración, Snort no se puede situar en múltiples interfaces a partir de una

única instalación, por lo que, solo es posible capturar en una instancia de un nodo al mismo tiempo. La figura 28 ejemplifica esta solución.

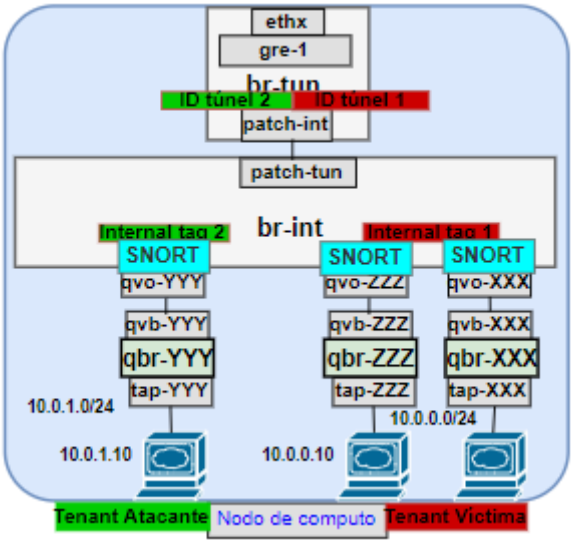

Figura 28. Solución nube privada en el nodo de cómputo.

5 Análisis de una instalación all-in-one.

En un principio se realiza un análisis detallado de una implementación de Openstack distribuida en múltiples nodos físicos. Sin embargo, no se cuenta con el presupuesto ni los recursos suficientes para llevar a cabo una instalación multinodo de una nube privada Openstack. Los desarrolladores han puesto a disposición de los usuarios una herramienta que permite instalar los servicios esenciales de Openstack en un único nodo. Se trata de un entorno de pruebas configurable mediante un escueto fichero previo a su instalación. En este proyecto se opta por implementar este entorno de pruebas conocido como all-in-one. Se dedica esta sección a exponer un análisis de las características básicas de este tipo de instalación.

En base al objetivo del proyecto el análisis se centra en el servicio de red Neutron. Se trata de afianzar los conceptos expuestos hasta el momento con un análisis aplicado de forma práctica a una implementación all-in-one. Se muestra la redistribución de los agentes y componentes implicados en el servicio de red, así como, las situaciones de vulnerabilidad y localizaciones de un NIDS extrapoladas a este tipo de instalación.

5.1 Agentes y servicios de Neutron

Evidentemente la distribución de los agentes y servicios no es idéntica a la de una instalación multinodo. Sin embargo, su funcionamiento es prácticamente el mismo, facilitando presentar las características de esta implementación en base a los conocimientos adquiridos en el estudio de una instalación completa multinodo.

Entre los servicios y agentes de Neutron presentes en este despliegue se encuentran todos los estudiados hasta el momento, con la única diferencia de que se encuentran todos corriendo en el mismo host localhost.localdomain como se puede ver en la figura 29.

```
[root@localhost ~(keystone_admin)]# neutron agent-list
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.
+--------------------------------------+--------------------+-----------------------+-------------------+
| id                                   | agent_type         | host                  | availability_zone |
+--------------------------------------+--------------------+-----------------------+-------------------+
| 39c2409b-ba50-4ab2-bb1b-01de9ce8d6e8 | Open vSwitch agent | localhost.localdomain |                   |
| 3c0c835e-a30b-4d25-9d63-4cab69505acd | Metadata agent     | localhost.localdomain |                   |
| 6d587cf3-6560-4cd4-b4a3-10d5355391d2 | DHCP agent         | localhost.localdomain | nova              |
| 8ca127c7-cd6e-448c-9c0c-b1be6943956e | L3 agent           | localhost.localdomain | nova              |
| fcf2bc90-15b2-4772-8453-63a8e6c8d731 | Metering agent     | localhost.localdomain |                   |
+--------------------------------------+--------------------+-----------------------+-------------------+
```

Figura 29. Agentes y servicios de Neutron activos en una instalación all-in-one.

5.2 Componentes de Neutron

En cuanto a los componentes que forman la red virtual en el interior del nodo, se encuentran los tres bridges OpenvSwitch, los equipos Linux bridge, las interfaces para el reenvió y recepción de datos, y los espacios de nombre, tanto de red como del servicio dhcp.

5.2.1 Bridges OpenvSwitch

Se presenta la estructura interna de los bridges OpenvSwitch con ayuda de las figuras 30, 31, 32, 33 y 34. La instalación cuenta con una red de proveedor externa, un router virtual, una red tenant y una instancia con el fin de que se encuentren el máximo número de componentes en la explicación.

El bridge de integración cuenta con seis puertos, dos para la conexión con el resto de bridges (1,2), tres para implementar los dos espacios de nombres correspondientes a un solo tenant (3,4,6) y uno para cada instancia conectada(5).

```
[root@localhost ~]# ovs-ofctl show br-int
OFPT_FEATURES_REPLY (xid=0x2): dpid:00007e2d24f66d4c
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst
 1(int-br-ex): addr:3a:0f:07:72:8a:df
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 2(patch-tun): addr:82:00:90:8d:40:8e
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 3(qr-03dd703a-5c): addr:00:00:00:00:00:00
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
 4(qg-17d482dd-b7): addr:00:00:00:00:00:00
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
 5(tapf7bd4f9f-08): addr:00:00:00:00:00:00
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
 6(qvo94ea42bc-bf): addr:8e:5c:f7:b1:fc:02
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(br-int): addr:7e:2d:24:f6:6d:4c
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Figura 30. Puertos del bridge de integración.

El bridge túnel cuenta con un puerto para la conexión con el bridge de integración. Al no existir una infraestructura de red física interna, este bridge permanece inutilizado en esta implementación.

```
[root@localhost ~]# ovs-ofctl show br-tun
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000d21636fbb14a
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst
 1(patch-int): addr:3a:f4:94:1d:91:03
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(br-tun): addr:d2:16:36:fb:b1:4a
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Figura 31. Puertos del bridge túnel.

El bridge external cuenta con dos puertos, uno conectado a la interfaz física del nodo (1), enp0s8, y otro para la conexión con el bridge de integración (2).

```
[root@localhost ~]# ovs-ofctl show br-ex
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000080027375ec1
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst
 1(enp0s8): addr:08:00:27:37:5e:c1
     config:     0
     state:      0
     current:    1GB-FD COPPER AUTO_NEG
     advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
     supported:  10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
     speed: 1000 Mbps now, 1000 Mbps max
 2(phy-br-ex): addr:82:b1:86:6e:f9:16
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(br-ex): addr:08:00:27:37:5e:c1
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Figura 32.Puertos del bridge external.

Espacios de nombres

La instalación configura por defecto una simple topología de red para un tenant llamado demo. Esta topología cuenta con un servidor dhcp y con un router virtual para ofrecer conectividad externa a la red privada. Openstack hace uso de los siguientes espacios de nombres para cubrir estos servicios.

```
[root@localhost ~]# ip netns
qdhcp-b3599300-f0f9-4828-9342-2fe3e30d9ea4 (id: 1)
qrouter-ef4081f8-b602-43d5-a33e-287a1c694d02 (id: 0)
```

Figura 33. Espacios de nombres.

Equipos Linux Bridge

Se arranca una instancia con el objetivo de obtener una explicación más completa. Se crea automáticamente el correspondiente Linux Bridge donde se implementa el grupo de seguridad asignado a dicha instancia.

```
[root@localhost ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
qbr94ea42bc-bf          8000.02800ec4cadf       no              qvb94ea42bc-bf
                                                                tap94ea42bc-bf
```

Figura 34. Puertos del Linux Bridge.

5.3 Escenarios de vulnerabilidad

Con todos los componentes descritos y ejecutándose en el mismo nodo se presenta en la figura 35 la distribución de los componentes en el nodo. Con la estructura interna del nodo definida es más fácil realizar un análisis en busca de los principales puntos de vulnerabilidad del sistema.

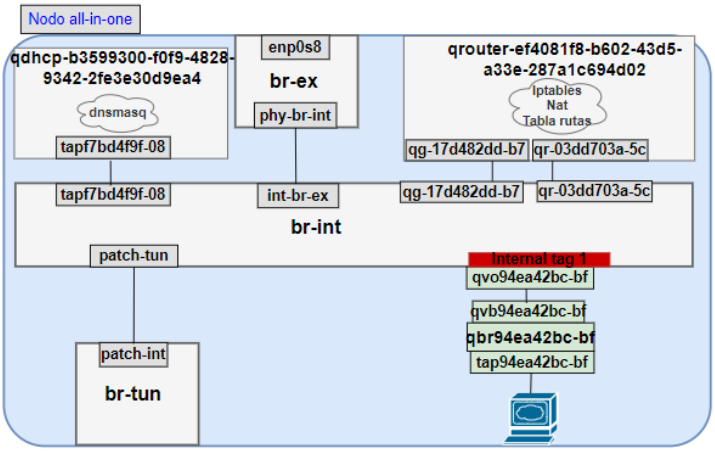

Figura 35. Estructura interna del nodo con instalación all-in-one.

El hecho de contar con un único nodo permite descartar los escenarios en los que se presupone un empleado deshonesto o un intruso con base en la nube situado en la misma subred, en otra subred del mismo tenant u otra red tenant en un nodo distinto de su objetivo.

El resto de los escenarios continúan presentes en este tipo de instalación por lo que se realiza un breve repaso.

5.3.1 Origen externo

Se trata de una situación prácticamente idéntica a la presentada en la infraestructura multinodo. El camino de datos se ve modificado ante la ausencia de red física. Sin embargo, la función de cada componente continua inalterada, exceptuando al bridge túnel que no cumple ninguna función en este tipo de instalación. El trayecto entre los bridges de integración de ambos nodos se simplifica en uno solo.

5.3.2 Origen interno

Se mantienen las tres posibilidades restantes en las que se presupone un empleado deshonesto o un intruso asentado en la nube.

- La situación de un empleado deshonesto situado en otra red tenant o situado en otra subred de la misma red tenant de su objetivo. Ambas situaciones comparten gran parte del camino de datos.
- La situación de un empleado deshonesto en la misma subred de su objetivo. Los datos se conmutan de una instancia a otra directamente en el bridge de integración sin pasos intermedios. En este escenario el camino de datos no se ve alterado con respecto a una infraestructura multinodo.

5.4 Localizaciones del NIDS

Del mismo modo que en los escenarios de vulnerabilidad, se puede descartar la localización del NIDS en un nodo dedicado de la infraestructura multinodo.

Las otras dos configuraciones mantienen sus principales características, variando en el perjuicio causado sobre la actuación de la nube. La primera se divide en dos posibles localizaciones manteniendo sus características intactas. La tercera configuración pasa a tener un impacto importante en el rendimiento de la nube, puesto que antes, perjudicaba el rendimiento del nodo de cómputo y ahora a la nube en su conjunto. Se presentan las tres localizaciones de Snort correspondientes a ambas configuraciones en la figura 36.

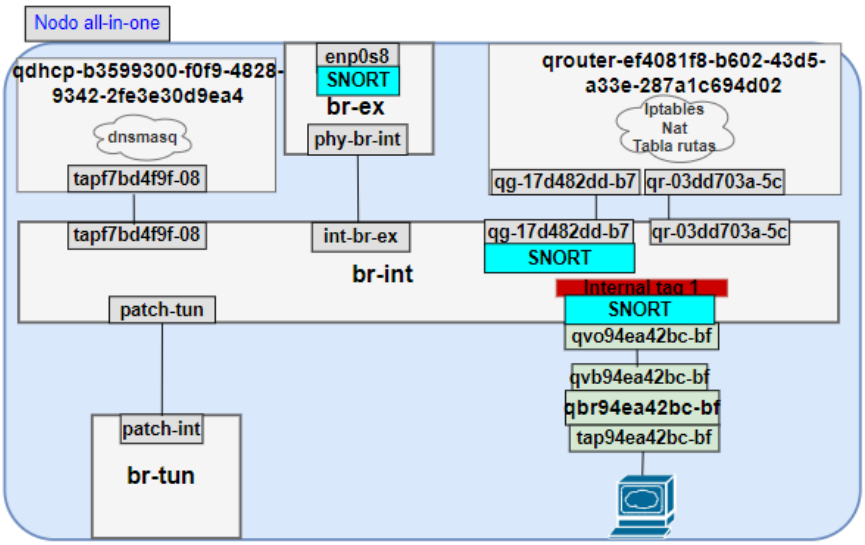

Figura 36. Solución nube privada con instalación all-in-one.

6 Desarrollo del proyecto

Una vez se encuentran expuestos los conceptos relacionados con la detección y prevención de intrusos en un entorno de virtualización como lo es Openstack, se procede a exponer el desarrollo práctico llevado a cabo como complemento a los escenarios presentados de forma teórica.

Para ello se reproduce el siguiente entorno en una plataforma de virtualización: una empresa con dos departamentos con una nube privada accesible a través de internet para sus empleados.

Partiendo de este entorno, se recrean las situaciones de intrusión y la solución que un administrador de nube privada puede implementar ante estos escenarios. La solución consta de una arquitectura de seguridad basada en un sistema de detección de intrusos como Snort y un sistema para el filtrado de tráfico, ya sea mediante grupos de seguridad o la extensión de Neutron Firewall as a Service.

6.1 Entorno hardware

El rendimiento de un entorno Openstack está directamente relacionado a la capacidad del recurso físico sobre el que se implementa. Se dedica este apartado a definir las características físicas de la máquina finalmente seleccionada, así como, una breve reseña de las limitaciones hardware encontradas durante la implementación del proyecto.

6.1.1 Especificaciones hardware

El trabajo se termina realizando sobre un ordenador portátil del fabricante Acer con un procesador Intel(R)Core(TM) i5-7200U de 2,5GHz, 8G de memoria RAM y un disco SSD de 200 GB. El sistema operativo instalado sobre el equipo es Windows 10 de 64 bits.

6.1.2 Problemas hardware

En un principio se pretende realizar el proyecto sobre un ordenador portátil del fabricante ASUS con un procesador Intel(R) Core(TM) i5-3337U con CPU de 1,8GHz, 8GB de memoria RAM y un disco duro HDD de 1TB.

Se trata de un sistema basado en una arquitectura de 64 bits, pero cuenta con un sistema operativo Windows 7 de 32 bits, por lo que, la memoria RAM está limitada a 4GB.

La página oficial del proyecto RDO recomienda utilizar un dispositivo físico con al menos 16GB de memoria RAM junto con un sistema operativo Centos7 o equivalentes de otras distribuciones soportadas. La única arquitectura soportada hasta el momento es de 64 bits.

Con el objetivo de aproximar la capacidad del dispositivo lo máximo posible a los prerrequisitos establecidos, se decide llevar a cabo una partición. En una partición todos los sistemas operativos deben basarse en la misma arquitectura, por lo que, se actualiza el sistema operativo actual a Windows 10 de 64 bits.

La principal complicación a la hora de realizar una partición del disco duro surge de la necesidad de que ambos sistemas sean instalados con el mismo tipo de arranque desde la BIOS.

La BIOS cuenta con dos tipos de arranque, uno antiguo conocido como Legacy y su sustituto en la actualidad, el arranque UEFI. Antes de realizar la partición es importante conocer el tipo de arranque utilizado durante la instalación del sistema operativo preinstalado de fábrica. La información del sistema permite conocer este dato.

El dispositivo en cuestión cuenta con un sistema Windows preinstalado de fábrica con el modo de arranque Legacy. Esto implica que el sistema operativo CentOS7 debe ser instalado con la BIOS configurada en este modo de arranque.

La instalación de CentOS7 sobre la partición se realiza desde un dispositivo USB convertido previamente en un dispositivo de arranque. Se configura la BIOS para arrancar el dispositivo USB en modo Legacy y se procede a la instalación. El último paso para obtener un dual boot consiste en configurar el fichero de arranque característico de los sistemas Linux, grub2.

Una vez finalizado el proceso, se obtiene un equipo que tras cargar el firmware BIOS ofrece al usuario la posibilidad de elegir el sistema operativo sobre el que arrancar.

El entorno Openstack se implementa sobre la partición con CentOS7, por lo que, se consigue la máxima dedicación de recursos como el procesador y la memoria RAM.

Sin embargo, los resultados no fueron los esperados. Se intentó con diferentes versiones de Openstack, pero la instalación finalizaba con errores o incompleta. Además, la lentitud y el retraso en la respuesta del equipo imposibilitaba cualquier intento de continuar el proyecto así.

Ante estos resultados, se intentó realizar el proyecto sobre el sistema operativo Windows 10, aprovechando softwares de virtualización como Virtual Box para generar una máquina Centos7. Aunque en este caso la velocidad de respuesta del equipo aumento notablemente, la instalación continúo arrojando los mismos problemas.

6.2 Entorno de virtualización

Para realizar la virtualización de todo el sistema podría haberse utilizado cualquiera de las herramientas comerciales existentes, como son VMWare o Hyper-V, pero dado que el IDS utilizado es opensource, se mantiene dicha filosofía también en la elección del sistema de virtualización, por lo que se ha utilizado VirtualBox.

6.2.1 Entorno Virtual Box

La herramienta utilizada para la instalación de Openstack es soportada por las distribuciones de Linux, CentOS7, Fedora y Red Hat Enterprise. Este requisito junto al hecho de que el equipo físico cuente con un sistema Windows induce la necesidad de utilizar un entorno de virtualización que permita generar una máquina con alguno de estos sistemas.

Virtual Box es un software de virtualización que permite satisfacer esta necesidad y obtener beneficio de otras características. Ofrece la posibilidad de generar una red virtual interna en el anfitrión mediante la virtualización de la tarjeta de red física.

En esta red el anfitrión cuenta con un adaptador virtual, conocido como adaptador solo-anfitrión, con la dirección ip 192.168.56.1 por defecto. Cualquier máquina virtual en la red tiene un adaptador del tipo VirtualBox Host-Only Ethernet Adapter habilitado.

Aprovechando las características de esta red, se decide implementar Openstack sobre una máquina CentOS7 con un adaptador VirtualBox Host-Only habilitado. De esta forma, se consigue un laboratorio portátil, implementado sobre una red fija que no depende de la red LAN a la que se conecta el equipo físico. Se evita implementar la nube sobre otro adaptador por comodidad a la hora de trabajar.

Es el caso de los adaptadores, se opta por utilizar el adaptador en modo puente, esto es, la dirección IP de la máquina virtual es asignada por el servidor DHCP del gateway de la red LAN a la que se conecte el equipo en cada momento, en vez de hacer uso del adaptador en modo NAT, ya que requeriría la configuración del reenvió de puertos para acceder a los distintos servicios de Openstack desde el anfitrión.

Se habilita otro adaptador tipo NAT secundario, para tener acceso a Internet y poder realizar las descargas y actualizaciones de paquetes necesarias durante la instalación. Otra posibilidad sería modificar la tabla de rutas del anfitrión, añadiendo la ruta por defecto, es decir, la de salida a internet. En esta ruta se configura tanto la puerta de enlace como la interfaz de salida con la dirección ip del router de la LAN y con el adaptador solo anfitrión respectivamente. La tabla de rutas quedaría como se aprecia en la figura 37.

```
IPv4 Tabla de enrutamiento
===========================================================================
Rutas activas:
Destino de red        Máscara de red   Puerta de enlace   Interfaz  Métrica
          0.0.0.0    255.255.255.0      192.168.1.1     192.168.56.1     26
        127.0.0.0        255.0.0.0      En vínculo         127.0.0.1    331
        127.0.0.1  255.255.255.255      En vínculo         127.0.0.1    331
  127.255.255.255  255.255.255.255      En vínculo         127.0.0.1    331
      192.168.1.0    255.255.255.0      En vínculo     192.168.1.157    291
    192.168.1.157  255.255.255.255      En vínculo     192.168.1.157    291
    192.168.1.255  255.255.255.255      En vínculo     192.168.1.157    291
     192.168.56.0    255.255.255.0      En vínculo      192.168.56.1    281
     192.168.56.1  255.255.255.255      En vínculo      192.168.56.1    281
   192.168.56.255  255.255.255.255      En vínculo      192.168.56.1    281
        224.0.0.0        240.0.0.0      En vínculo         127.0.0.1    331
        224.0.0.0        240.0.0.0      En vínculo      192.168.56.1    281
        224.0.0.0        240.0.0.0      En vínculo     192.168.1.157    291
  255.255.255.255  255.255.255.255      En vínculo         127.0.0.1    331
  255.255.255.255  255.255.255.255      En vínculo      192.168.56.1    281
  255.255.255.255  255.255.255.255      En vínculo     192.168.1.157    291
===========================================================================
```

Figura 37. Tabla de rutas del equipo anfitrión.

Se obtiene una topología de red como la mostrada en la figura 38. Una máquina virtual con dos adaptadores de red. Uno para implementar la nube Openstack sobre una red fija y otro para tener conectividad con internet a través de un router conectado directamente a la tarjeta de red del equipo físico.

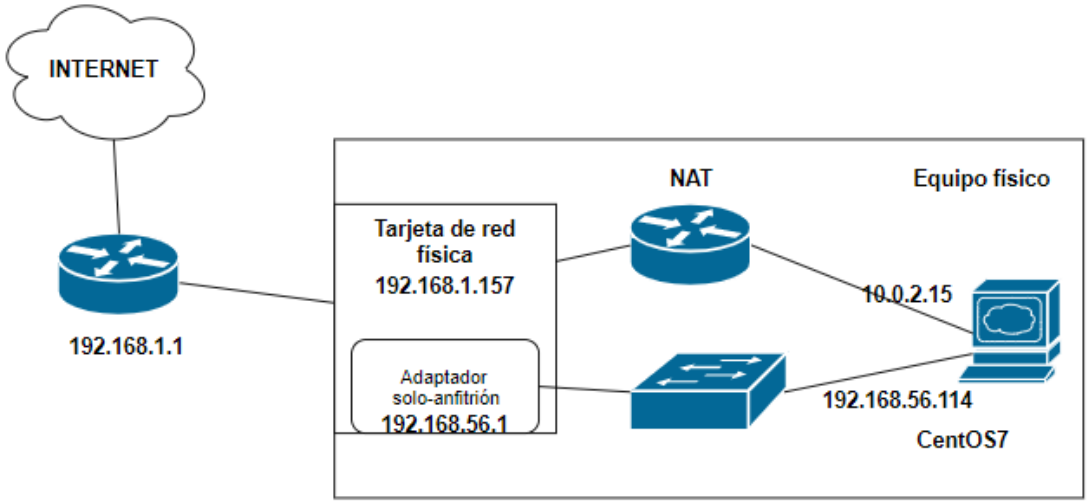

Figura 38. topología de red con entorno Virtual Box.

Otras características destacables de la máquina virtual CentOS7 están relacionadas al almacenamiento y la capacidad del sistema.

Se asigna un disco duro virtual reservando un espacio fijo de 30 GB. Se trata de una cantidad de memoria escasa para la implementación de una nube, pero no se dispone de más espacio. En cuanto a las características del sistema se reservan 5171 MB de memoria RAM de los 8000 MB disponibles. Además, se asignan dos CPUs virtuales del procesador.

6.2.2 Entorno Openstack

Se decide utilizar Packstack como instalador all-in-one de Openstack. Se trata de una herramienta más estable que otras con el mismo objetivo como Devstack. Se encuentra en constante actualización y mantenimiento por la comunidad del proyecto RDO, formada por desarrolladores de Openstack sobre distribuciones CentOS, Fedora o Red Hat Enterprise.

Packstack permite la instalación de Openstack en un único nodo sin realizar ningún tipo de configuración mediante la opción –allinone o a través de un fichero de configuración modificable que se pasa como parámetro de la opción --answer-file.

La última versión estable de Openstack es la conocida como Stein. Sin embargo, por problemas relacionados con la capacidad del dispositivo físico es necesario reducir las pretensiones exigidas por dicha versión de Openstack. Se decide utilizar la versión Queens cuyos requisitos más flexibles permiten terminar la instalación sin errores.

Se modifica el fichero de configuración antes de proseguir con la instalación. Se actualiza la dirección ip de acceso a los servicios de la nube de acuerdo con el adaptador VirtualBox Host-Only de la máquina CentOS7, la contraseña del usuario admin con el fin de facilitar el acceso al dashboard y otros parámetros relacionados con el servicio Neutron.

Entre estos parámetros se encuentran opciones como el plugin del Neutron-L2-plugin-Agent, los drivers del plugin, el mecanismo para la virtualización de la red tenant y de la red proveedor, el nombre de la red externa, las extensiones de Neutron o la interfaz a mapear sobre el bridge external.

```
CONFIG_LBAAS_INSTALL=n
CONFIG_NEUTRON_METERING_AGENT_INSTALL=y
CONFIG_NEUTRON_FWAAS=y
CONFIG_NEUTRON_VPNAAS=n
CONFIG_NEUTRON_ML2_TYPE_DRIVERS=vxlan,flat
CONFIG_NEUTRON_ML2_TENANT_NETWORK_TYPES=vxlan
CONFIG_NEUTRON_ML2_MECHANISM_DRIVERS=openvswitch
CONFIG_NEUTRON_ML2_VNI_RANGES=10:100
CONFIG_NEUTRON_L2_AGENT=openvswitch
CONFIG_NEUTRON_ML2_SUPPORTED_PCI_VENDOR_DEVS=['15b3:1004', '8086:10ca']
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=extnet:br-ex
CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:enp0s8
CONFIG_NEUTRON_OVS_EXTERNAL_PHYSNET=extnet
```

Figura 39. Configuración del fichero answer.txt.

En la figura 39 se muestra la configuración seleccionada. Se activa la extensión firewall as a service, se selecciona OpenvSwitch como driver del plugin modular layer 2 y se mapea la interfaz enp0s8 de la máquina CentOS7 en el br-ex de OpenvSwitch. Los mecanismos para la virtualización de redes Tenant y redes de Proveedor, vxlan y flat respectivamente, no tienen mayor relevancia al tratarse de una implementación all-in-one.

6.3 Dominio del Administrador

La instalación se completa con un entorno constituido por un usuario admin y un usuario demo. El primero permite al administrador de la nube modelar el entorno a su agrado. En su poder se encuentra dar acceso a los usuarios de la nube y definir el modelo a seguir para entregar la red como servicio. El segundo es un usuario de prueba que se crea por defecto. En esta implementación se elimina con el objetivo de preservar el máximo espacio posible.

El usuario admin pertenece a un proyecto denominado admin. Por defecto, tras la instalación, se incluye una red denominada public asociada a dicho proyecto. Los atributos external y compartido de la red se sitúan como verdadero y falso respectivamente, tal y como se aprecia en la figura 40.

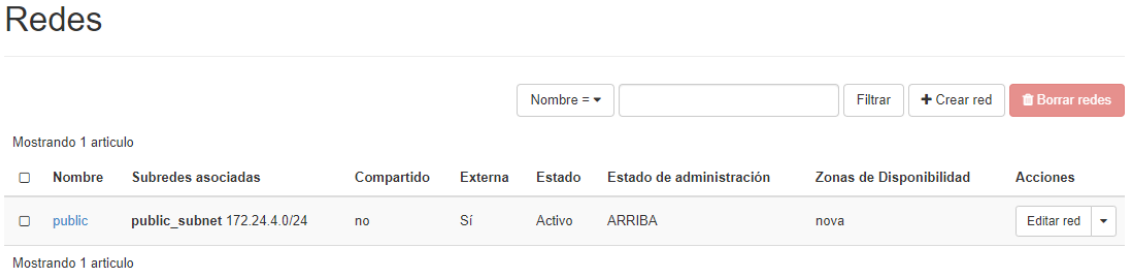

Figura 40. Características de la red public.

La red public se enmarca en las redes de proveedor concretamente la conocida como red externa. Se trata de una red típicamente utilizada en los entornos Openstack bajo un modelo self-service. Cada tenant tiene acceso al exterior de la nube mediante un router virtual en contacto directo con esta red.

Se decide continuar la implementación aprovechando las características del modelo self-service y utilizando como red externa la red public.

El privilegio de registrar nuevos usuarios le corresponde de nuevo al usuario admin. Se decide generar desde la CLI con las credenciales del usuario admin dos nuevos usuarios, UserA y UserB, y dos nuevos proyectos, TenantA y TenantB. Además, se asigna un rol a estos usuarios en sus respectivos proyectos, en ambos casos se asigna el rol preestablecido _member_ tal y como se aprecia en la figura 41.

```
[root@localhost ~(keystone_admin)]# openstack role assignment list --role _member_ --names
+----------+---------------+-------+-----------------+--------+-----------+
| Role     | User          | Group | Project         | Domain | Inherited |
+----------+---------------+-------+-----------------+--------+-----------+
| _member_ | UserA@Default |       | TenantA@Default |        | False     |
| _member_ | UserB@Default |       | TenantB@Default |        | False     |
+----------+---------------+-------+-----------------+--------+-----------+
```

Figura 41. Asociación de roles en pares usuario-proyecto.

Con dos usuarios asociados a sus respectivos proyectos es suficiente para reproducir todos los escenarios estudiados.

6.4 Dominio del Usuario

Cada llamada a la API de Openstack requiere de las credenciales del usuario en cuestión, por lo que, Openstack permite descargar un fichero con estas credenciales para cada usuario registrado. Ejecutándose como fuente permite eliminar este problema. Por comodidad, es más fácil crear un par de copias del fichero perteneciente al usuario admin y adaptarlo a los usuarios UserA y UserB. En la figura 42 se muestra la configuración realizada para el UserA.

```
[root@localhost ~(keystone_admin)]# cat keystonerc_UserA
unset OS_SERVICE_TOKEN
export OS_USERNAME=UserA
export OS_PASSWORD='TenantA'
export PS1='[\u@\h \W(keystone_UserA)]\$ '
export OS_AUTH_URL=http://192.168.56.114:5000/v3

export OS_PROJECT_NAME=TenantA
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IDENTITY_API_VERSION=3
```

Figura 42. Configuración del fichero Keystonerc_UserA.

Una vez se encuentran registrados en una nube con un modelo self-service, ambos usuarios adquieren suficientes privilegios para diseñar una topología de red que se adapte a sus necesidades.

Con objetivo de preservar espacio suficiente, se diseña una topología de red sencilla para cada tenant, pero que permita recrear los escenarios previamente estudiados.

Cada tenant cuenta con un router virtual, una red privada y una instancia en dicha red. A continuación, se describe el proceso de creación seguido por cualquier usuario:

- Se genera la red privada con la subred asociada.
- Se arranca una instancia cirros sobre la red privada.
- Se crea un router virtual.
- Se añade el gateway de la red externa al router.
- Se añade una interfaz de la red privada al router.
- Se asocia una dirección ip flotante a la instancia.

6.5 Diseño de los escenarios

Se presentan distintos escenarios en la nube privada recién generada. En cada escenario se recrea un intento de comunicación mediante el envió de un ping que genere una alerta en el sistema de detección de intrusos del administrador. Además, se propone la solución a cada situación.

6.5.1 Escenario con origen externo

El usuario UserA diseña una red con una instancia dentro del proyecto TenantA. La instancia únicamente cuenta con la propia protección de su firewall:

- Dirección ip de la red tenantA: 10.0.0.0/24
- Dirección ip privada de la instancia instanceA: 10.0.0.15.
- Dirección ip flotante de la instancia instanceA:172.14.4.11.

El administrador monitoriza la nube con un sistema de detección de intrusos en el bridge external.

La figura 43 muestra la topología de red del TenantA.

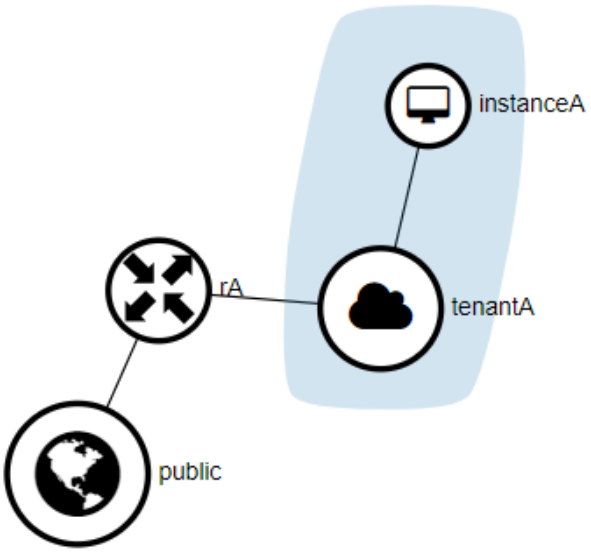

Figura 43. Topología de red en el proyecto TenantA.

Se realiza un ping desde el equipo anfitrión Windows a la dirección ip flotante de la instancia, de forma que, Snort avisa al administrador con las alertas que se muestran en la figura 44.

```
Commencing packet processing (pid=624)
09/27-19:19:47.920330  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.1 -> 172.24.4.11
09/27-19:19:49.008652  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.1 -> 172.24.4.11
09/27-19:19:50.097385  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.1 -> 172.24.4.11
09/27-19:19:51.219649  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.1 -> 172.24.4.11
^C*** Caught Int-Signal
===============================================================================
Run time for packet processing was 460.477168 seconds
Snort processed 1207 packets.
Snort ran for 0 days 0 hours 7 minutes 40 seconds
   Pkts/min:          172
   Pkts/sec:            2
===============================================================================
```

Figura 44. Alertas de Snort situado en el br-ext.

Las alertas son generadas por la siguiente regla snort incluida en fichero local.rules:

alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)

La variable $HOME_NET contiene la dirección de la red que desea proteger el administrador, en este caso, la dirección 172.24.4.0/24 de la red public.

La petición se lanza con la dirección origen 192.168.56.1 del adaptador virtual generado por virtual box. Antes de entrar en la red public se intercambia la dirección origen por la del gateway de la red public. La respuesta no genera ninguna alerta de Snort, puesto que va dirigida a la red 192.168.56.0/24 que no se encuentra protegida por la regla definida.

El usuario no ha definido seguridad más allá de la que proporciona el propio firewall de la instancia, por lo que el ping se resuelve correctamente como se muestra en la figura 45.

```
C:\Users\USUARIO>ping 172.24.4.11

Haciendo ping a 172.24.4.11 con 32 bytes de datos:
Respuesta desde 172.24.4.11: bytes=32 tiempo=6ms TTL=62
Respuesta desde 172.24.4.11: bytes=32 tiempo=5ms TTL=62
Respuesta desde 172.24.4.11: bytes=32 tiempo=5ms TTL=62
Respuesta desde 172.24.4.11: bytes=32 tiempo=34ms TTL=62

Estadísticas de ping para 172.24.4.11:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 5ms, Máximo = 34ms, Media = 12ms
```

Figura 45. Resolución del ping desde el anfitrión.

6.5.2 Escenario con origen interno

En este caso, a su vez, se distinguen tres subescenarios: misma subred de un tenant, distinta subred del mismo tenant o distinta red tenant.

6.5.2.1 Misma subred de un Tenant

El usuario UserA diseña una red con dos instancias, instanceA e instanceA1, dentro del proyecto TenantA, las instancias únicamente cuentan con la propia protección de su propio firewall:

- Dirección ip de la red tenantA: 10.0.0.0/24
- Dirección ip privada de la instancia instanceA: 10.0.0.15.
- Dirección ip privada de la instancia instanceA1: 10.0.0.11.

El administrador monitoriza la nube con un sistema de detección de intrusos en la interfaz qvo-XXX correspondiente a la instancia instanceA1 situada en el bridge de integración.

La figura 46 muestra la topología de red del TenantA.

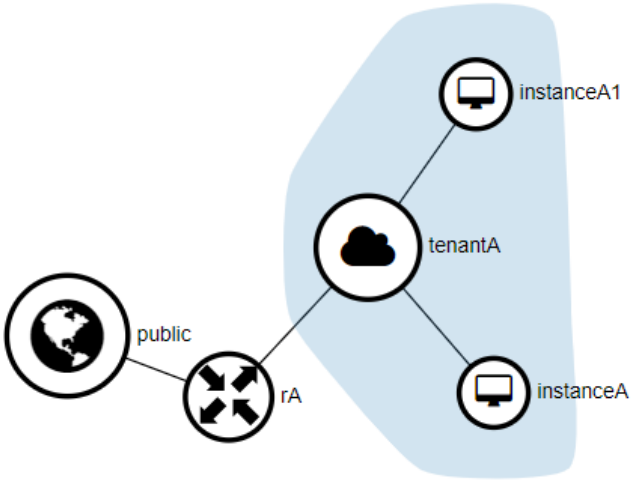

Figura 46. Topología de red del TenantA con dos instancias.

Se realiza un ping desde el equipo instanceA a la dirección ip privada de la instancia instanceA1, de forma que, Snort alerta al administrador con las alertas que se muestran en la figura 47.

```
Commencing packet processing (pid=4126)
09/27-16:19:33.436217  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.0.0.15 -> 10.0.0.11
09/27-16:19:33.439390  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.0.0.11 -> 10.0.0.15
09/27-16:19:34.436867  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.0.0.15 -> 10.0.0.11
09/27-16:19:34.438190  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.0.0.11 -> 10.0.0.15
09/27-16:19:35.439977  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.0.0.15 -> 10.0.0.11
09/27-16:19:35.440739  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.0.0.11 -> 10.0.0.15
^C*** Caught Int-Signal
===============================================================================
Run time for packet processing was 58.121881 seconds
Snort processed 8 packets.
Snort ran for 0 days 0 hours 0 minutes 58 seconds
   Pkts/sec:            0
===============================================================================
```

Figura 47. Alertas de Snort situado en la interfaz qvo del br-int.

Estas alertas son generadas por la siguiente regla de snort incluida en el fichero local.rules:

alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)

La variable $HOME_NET contiene la dirección de la red que desea proteger el administrador, en este caso, la dirección 10.0.0.0/24 de la red tenantA.

Las alertas corresponden tanto a la petición echo del ping como a su respuesta, puesto que se define la regla para todos los paquetes con destino a la red tenantA.

A pesar de las alertas generadas, el ping se resuelve sin problemas como se puede ver en la figura 48.

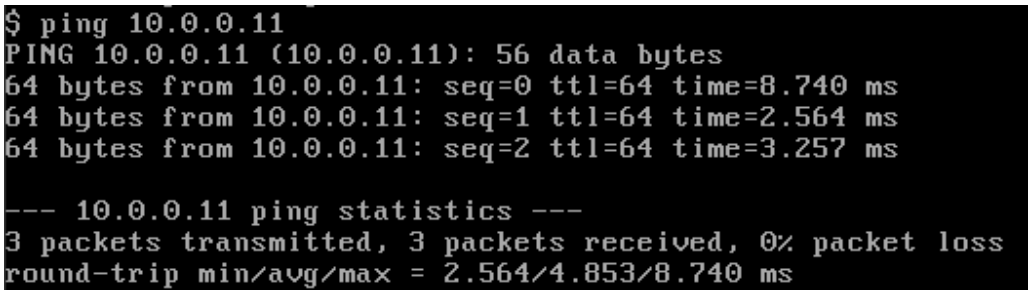

Figura 48. Resolución del ping desde una instancia.

6.5.2.2 Distinta subred en el mismo Tenant

El usuario UserA diseña dos subredes, departamentoA y departamentoB, dentro del proyecto TenantA. Cada subred cuenta con una instancia, instanceA e instanceB. Las instancias únicamente cuentan con la protección de su propio firewall:

- Dirección ip de la subred departamentoA: 10.0.0.0/24.
- Dirección ip de la subred departamentoB: 10.0.1.0/24.
- Dirección ip flotante de la instancia instanceA: 172.24.4.13.
- Dirección ip flotante de la instancia instanceB: 172.24.4.11.
- Dirección ip privada de la instancia instanceB: 10.0.0.14.

Hace falta monitorizar el tráfico en la interfaz pública del router para detectar cualquier actividad con anterioridad a su entrada en la red que se pretende proteger. Como esta interfaz se encuentra en un puerto interno del bridge de integración no es visible para Snort. Es necesario generar una interfaz accesible para Snort, adjuntarla a un puerto del bridge de integración y aprovechar la funcionalidad que ofrece OpenvSwitch para realizar port mirroring entre sus puertos.

El administrador realiza este procedimiento para la interfaz qg-XXX del router a monitorizar, de forma que, se analiza el tráfico entrante a cualquiera de las dos subredes.

La figura 49 muestra la topología de red del TenantA.

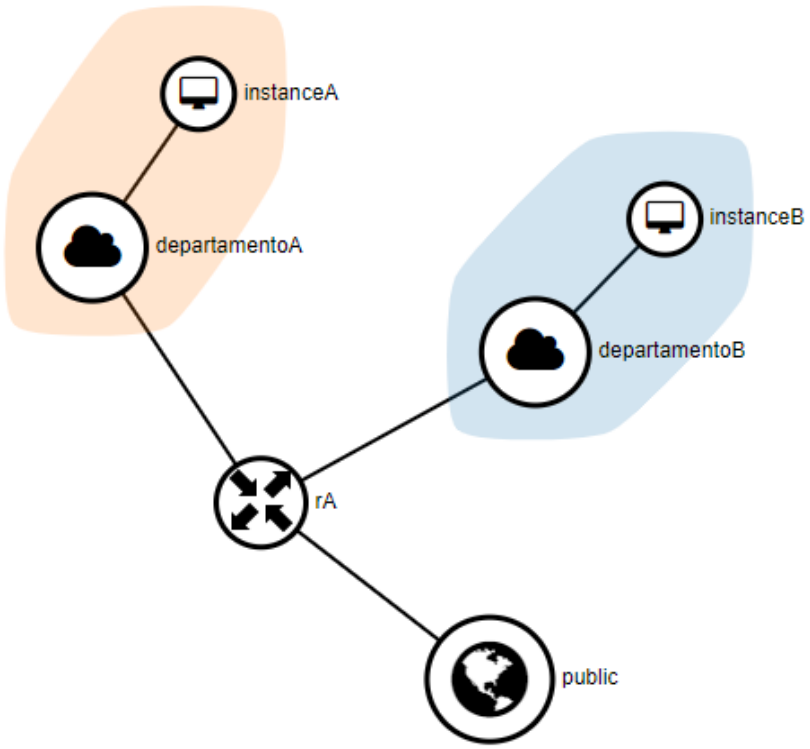

Figura 49. Topología de red con dos subredes.

Se realiza un ping desde el equipo instanceB a la dirección ip flotante de la instancia instanceA, de forma que, Snort alerta al administrador con las alertas que se muestran en la figura 50.

```
Commencing packet processing (pid=1329)
09/27-16:01:22.176463  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.1.14 -> 172.24.4.13
09/27-16:01:22.185599  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.14 -> 172.24.4.11
09/27-16:01:23.207037  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.1.14 -> 172.24.4.13
09/27-16:01:23.208326  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.14 -> 172.24.4.11
09/27-16:01:24.208566  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.1.14 -> 172.24.4.13
09/27-16:01:24.214402  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.14 -> 172.24.4.11
^C*** Caught Int-Signal
===============================================================================
Run time for packet processing was 69.195586 seconds
Snort processed 188 packets.
Snort ran for 0 days 0 hours 1 minutes 9 seconds
   Pkts/min:          188
   Pkts/sec:            2
===============================================================================
```

Figura 50. Alertas de Snort situado en la interfaz qg de router rA.

Estas alertas son generadas por la siguiente regla de Snort incluida en el fichero local.rules:

alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)

La variable $HOME_NET contiene la dirección de la red que desea proteger el administrador, en este caso, la dirección 172.24.4.0/24 de la red public.

En base a la regla definida se generan dos alertas por cada ping, uno para la petición procedente de la red 10.0.1.0/24 y otro para la respuesta procedente de la red 10.0.0.0/24. Ambos, petición y respuesta, llevan como destino la ip flotante de su instancia objetivo.

No se ha definido seguridad más allá de la que proporciona el propio firewall de la instancia, por lo que el ping se resuelve correctamente como se muestra en la figura 51.

```
$ ping 172.24.4.13
PING 172.24.4.13 (172.24.4.13): 56 data bytes

--- 172.24.4.13 ping statistics ---
11 packets transmitted, 0 packets received, 100% packet loss
$ ping 172.24.4.13
PING 172.24.4.13 (172.24.4.13): 56 data bytes
64 bytes from 172.24.4.13: seq=0 ttl=63 time=19.728 ms
64 bytes from 172.24.4.13: seq=1 ttl=63 time=2.168 ms
64 bytes from 172.24.4.13: seq=2 ttl=63 time=7.125 ms

--- 172.24.4.13 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.168/9.673/19.728 ms
```

Figura 51. Resolución del ping desde una instancia.

6.5.2.3 Distinta red tenant

En este escenario la nube alberga dos usuarios con sus respectivos proyectos. Se trata de un escenario aplicable a una empresa con múltiples departamentos. En este caso cada tenant o departamento cuenta con espacios de nombres independientes. Cada usuario diseña una red con una instancia dentro de su proyecto. La instancia únicamente cuenta con la propia protección de su firewall:

- Dirección ip de la red tenantA: 10.0.0.0/24.
- Dirección ip de la red tenantB: 10.0.0.0/24.
- Dirección ip flotante de la instancia instanceA: 172.24.4.4.
- Dirección ip flotante de la instancia instanceB: 172.24.4.12.
- Dirección ip privada de la instancia instanceA: 10.0.0.7.
- Dirección ip privada de la instancia instanceB: 10.0.0.11.

Del mismo modo que en el escenario anterior, se monitoriza el tráfico en la interfaz pública del router del tenant que se pretende proteger. Se decide proteger al TenantA, por lo que se realiza el port mirroring de la interfaz qg-XXX del router rA.

La figura 52 muestra la topología de red de la nube privada.

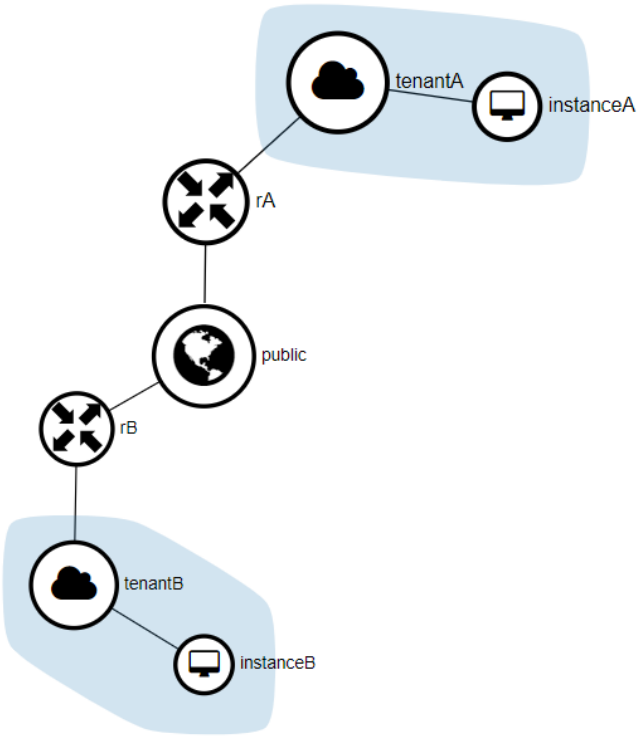

Figura 52. Topología de red con dos Tenant.

Se realiza un ping desde el equipo instanceB a la dirección ip flotante de la instancia instanceA, de forma que, Snort avisa al administrador con las alertas que se muestran en la figura 53.

```
Commencing packet processing (pid=819)
09/27-16:11:15.552715  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.11 -> 172.24.4.4
09/27-16:11:15.552879  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.12 -> 172.24.4.4
09/27-16:11:15.560290  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.7 -> 172.24.4.12
09/27-16:11:15.560750  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.4 -> 172.24.4.12
09/27-16:11:16.562563  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.11 -> 172.24.4.4
09/27-16:11:16.562754  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.12 -> 172.24.4.4
09/27-16:11:16.570117  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.7 -> 172.24.4.12
09/27-16:11:16.570187  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.4 -> 172.24.4.12
09/27-16:11:17.565090  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.11 -> 172.24.4.4
09/27-16:11:17.565351  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.12 -> 172.24.4.4
09/27-16:11:17.567711  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
0.0.0.7 -> 172.24.4.12
09/27-16:11:17.567786  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 1
72.24.4.4 -> 172.24.4.12
^C*** Caught Int-Signal
===============================================================================
Run time for packet processing was 190.710232 seconds
Snort processed 58 packets.
Snort ran for 0 days 0 hours 3 minutes 10 seconds
   Pkts/min:           19
   Pkts/sec:            0
===============================================================================
```

Figura 53. Alertas de Snort situado en la interfaz qg del router rA.

Las alertas son generadas por la siguiente regla snort incluida en fichero local.rules:

alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)

La variable $HOME_NET contiene la dirección de la red que desea proteger el administrador, en este caso, la dirección 172.24.4.0/24 de la red public.

Se generan cuatro alertas por cada ping, dos por la petición y dos por la respuesta. La primera en cada caso corresponde al paquete procedente de la instancia sin recibir modificación alguna por parte del router de su proyecto aún. La segunda representa al paquete una vez el router origen ha realizado el SNAT y lo ha reenviado al router destino.

El usuario no ha definido seguridad más allá de la que proporciona el propio firewall de la instancia, por lo que el ping se resuelve correctamente como se muestra en la figura 54.

```
$ ping 172.24.4.4
PING 172.24.4.4 (172.24.4.4): 56 data bytes
64 bytes from 172.24.4.4: seq=0 ttl=62 time=15.192 ms
64 bytes from 172.24.4.4: seq=1 ttl=62 time=13.264 ms
64 bytes from 172.24.4.4: seq=2 ttl=62 time=10.029 ms

--- 172.24.4.4 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 10.029/12.828/15.192 ms
```

Figura 54. Resolución del ping desde una instancia.

6.5.3 Soluciones

El administrador de la nube es quién recibe estas alertas, sin embargo, debe ser el usuario quién tome las acciones en el momento de proteger su red. En este contexto es evidente que debe existir comunicación entre usuario y administrador al detectarse actividad extraña en el tráfico de la red.

Entre los servicios que ofrece Openstack para detener tráfico se encuentran los grupos de seguridad, actuando como un firewall al nivel de instancias, y el firewall as a service, filtrando el tráfico a nivel de router.

Se presentan ambas soluciones aplicadas a los escenarios puestos en práctica.

- El firewall as service se aplica en aquellos escenarios en los que la red analizada por Snort corresponde a la red public. En estos casos es posible utilizar los grupos de seguridad, sin embargo, es preferible que el tráfico malintencionado interaccione lo menos posible con recursos del usuario, entre los que se incluye la red privada.

  En los escenarios puestos en práctica se crea una política de seguridad con una única regla que rechace el tráfico icmp con destino a la ip flotante de la instancia a proteger como se ve en la figura 55 de ejemplo.

```
[root@localhost ~(keystone_UserA)]# neutron firewall-rule-show reglaA
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.
+--------------------------+--------------------------------------+
| Field                    | Value                                |
+--------------------------+--------------------------------------+
| action                   | reject                               |
| description              |                                      |
| destination_ip_address   | 172.24.4.12                          |
| destination_port         |                                      |
| enabled                  | True                                 |
| firewall_policy_id       | 6f6ead36-13be-40f5-b58c-91e8e2066ce1 |
| id                       | 94af2c72-7391-4792-affe-6862d1e8f8d2 |
| ip_version               | 4                                    |
| name                     | reglaA                               |
| position                 | 1                                    |
| project_id               | 381cce207feb407aafc2a0043575c4a7     |
| protocol                 | icmp                                 |
| shared                   | False                                |
| source_ip_address        |                                      |
| source_port              |                                      |
| tenant_id                | 381cce207feb407aafc2a0043575c4a7     |
+--------------------------+--------------------------------------+
```

Figura 55. Regla en el firewall del router rA.

Dicha política se asocia a un firewall definido para el router del tenant protegido. A pesar del firewall, Snort continúa generando alertas, sin embargo, el ping no se resuelve con normalidad, termina agotándose el tiempo de espera como se muestra en la figura 56.

```
C:\Users\USUARIO>ping 172.24.4.4

Haciendo ping a 172.24.4.4 con 32 bytes de datos:
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.

Estadísticas de ping para 172.24.4.4:
    Paquetes: enviados = 4, recibidos = 0, perdidos = 4
    (100% perdidos),
```

Figura 56. Ping rechazado por el router rA.

- Los grupos de seguridad son aplicables con gran utilidad en el escenario con una única subred dentro del mismo tenant. En este caso los datos no atraviesan ningún router, por lo que no es aplicable el firewall as a service.

  Existe una gran variedad de opciones a la hora de definir una regla, en este ejemplo es suficiente con prohibir el tráfico icmp con destino a la red privada del escenario. De esta forma se consigue el mismo resultado para el ping que en la figura 56.

Existe la posibilidad de configurar Snort en modo activo, es decir, que permita la definición de reglas que no solo generen alertas, sino que actúen sobre las detecciones encontradas. Bajo esta perspectiva, el administrador es capaz de encontrar y detener las actividades que considere peligrosas sin necesidad de colaboración alguna por parte del usuario.

Se prueba esta posibilidad declarando la siguiente regla en el fichero de configuración local.rules:

reject icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)

Se sitúa Snort en el bridge external del escenario con origen externo y se genera un ping a la dirección ip flotante de la instancia protegida con el resultado de la figura 57.

```
C:\Users\USUARIO>ping 172.24.4.11

Haciendo ping a 172.24.4.11 con 32 bytes de datos:
Respuesta desde 172.24.4.11: Puerto de destino inaccesible.
Respuesta desde 172.24.4.11: Puerto de destino inaccesible.
Respuesta desde 172.24.4.11: Puerto de destino inaccesible.
Respuesta desde 172.24.4.11: Puerto de destino inaccesible.

Estadísticas de ping para 172.24.4.11:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
```

Figura 57. Ping rechazado por Snort.

Snort continúa generando alertas mientras se rechaza la petición del ping.

7 Conclusiones y líneas futuras

Los niveles de seguridad propuestos por los desarrolladores de Openstack centran sus esfuerzos en proteger la red desde sus extremos mediante herramientas especializadas en el filtrado de tráfico. Sin embargo, los administradores carecen de mecanismos para el control de la actividad que tiene lugar en el interior de la red. Por ello, resulta beneficioso explorar el uso de herramientas como Snort diseñadas para este tipo de servicios.

La responsabilidad de diseñar una arquitectura de seguridad confiable en un entorno de nube privada recae en la propia empresa propietaria de la infraestructura física, por lo que el administrador de la nube, encargado de la seguridad de la red, necesita conocer sobradamente el diseño y comportamiento del proyecto asociado al servicio de red en Openstack.

En este marco el trabajo presenta ampliamente las características y funcionamiento de Neutron, recreando escenarios aplicables a un entorno profesional, permitiendo conocer con exactitud el camino de datos recorrido por cualquier tipo de tráfico que pretenda alcanzar un recurso de la nube.

La premisa inicial del trabajo pretende realizar un análisis en profundidad de los factores que influyen a un administrador de una nube privada en la toma de decisión sobre el lugar donde localizar su sistema de detección de intrusos. Sin embargo, la escasez de presupuesto para implementar un entorno Openstack multinodo por un lado y la limitación hardware del recurso utilizado para implementar el entorno monomodo por otro, suponen un enorme hándicap en el análisis de estos factores.

Se ha modificado la premisa inicial para probar las distintas localizaciones del sistema de detección de intrusos enfocando el análisis del entorno al factor referido a los tipos de tráfico capturados. Se desprecian los análisis sobre el impacto en el rendimiento de la nube inferido en cada escenario, aunque de forma teórica se presentan de manera intuitiva los posibles resultados de cada uno de estos.

El trabajo evidencia la importancia que tiene el origen de las amenazas a la hora de localizar el sistema de detección de intrusos. Se clasifican y se prueban cuatro tipos de escenarios en los que los recursos de la nube son vulnerables ante amenazas con la configuración del NIDS más adecuada en cada escenario, exponiéndose los puntos fuertes y débiles de cada configuración. No existe una localización perfecta que cubra todo tipo de tráfico, pero en base al origen de la amenaza y al recurso que se desea proteger es posible seleccionar una configuración con mayores beneficios que el resto.

La fuerte dependencia del entorno de virtualización Openstack con la capacidad del hardware utilizado durante el proyecto hace que la evolución y continuidad futura del

proyecto se encuentre comprometida. Se ha utilizado una versión antigua de Openstack debido a las altas exigencias de las más recientes, por lo que parece evidente que a medida que se realicen nuevas actualizaciones continuarán incrementándose los prerrequisitos hardware del sistema. A medida que esto sucede, pierde cualquier sentido continuar realizando pruebas sobre versiones antiguas ante la evidente desactualización a la hora de reproducir estas sobre entornos reales.

El proyecto se ha visto limitado por distintos problemas relacionados con la capacidad del dispositivo, pero entre los más problemáticos se encuentra la limitación temporal impuesta por la escasa duración de las instancias en estado activo. Este problema impide realizar cualquier prueba con mayor elaboración que la requerida en un ping. Por tanto, se propone como futura línea de investigación, un estudio del impacto sobre el rendimiento del servidor, en términos de memoria y CPU utilizada, en un escenario concreto ante un ataque como la denegación de servicio o cualquier otro que evada el filtrado del firewall y se introduzca en el interior de la red, perjudicando el acceso o el funcionamiento normal de recursos como la red.

Otra posibilidad para futuros trabajos, algo más distante del proyecto actual, es introducirse en la gestión de alertas o logs generados por un sistema de detección de intrusos, en este caso Snort. Existen herramientas que facilitan la administración de estos sistemas mediante la centralización de eventos, ayuda en la interpretación de estos o el control sobre los sensores y la actualización de reglas en el sistema.

8 Referencias

[1] Mehmood, Y., Shibli, M. A., Habiba, U., & Masood, R. 2013. *Intrusion Detection System in Cloud Computing: Challenges and Opportunities*. Pakistán: National University of Science and Technology Islamabad, SEECS. Disponible en: https://www.researchgate.net/publication/271555430_Intrusion_Detection_System_in_Cloud_Computing_Challenges_and_opportunities.

[2] Rai, K., & Devi, M. S. (2013). *Intrusion Detection Systems: A Review*. India: Panjab University, Department of Computer Science and Applications. Vol. 1, Issue 2 .Disponible en: https://www.researchgate.net/publication/282274721_Intrusion_Detection_Systems_A_Review.

[3] Cantos Poliano, P. 2012. Snort. *Desarrollo de un sistema de búsqueda de patrones para prefiltrado de alta velocidad en Snort.* P. Nebrera Herrrera, director. Trabajo de fin de carrera, Universidad de Sevilla. Disponible en: http://bibing.us.es/proyectos/abreproy/12077/fichero/memoria%252FDocumento.pdf

[4]. Sánchez Lorente, O. 2015. Sistema de detección de intrusos. *Detección de intrusos con Snort.* C. Pérez Solà, directora. Trabajo de fin de grado, Universitat Oberta de Catalunya. Disponible en: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/43090/6/osanchezloTFM0715memoria.pdf

[5] Adminsos.es. *SNORT-Modos de ejecución – ASO.* Disponible en: http://www.adminso.es/index.php/SNORT-Modos_de_ejecuci%C3%B3n

[6] Seguridad en Sistemas y Técnicas de Hacking. TheHackerWay (THW). *Conceptos Básicos y Configuración de Snort – Parte I.* Disponible en: https://thehackerway.com/2011/07/01/conceptos-basicos-y-configuracion-de-snort-%e2%80%93-parte-i/

[7] Mirantis | Pure Play Open Cloud. *Blog | Mirantis.* Disponible en: https://www.mirantis.com/blog/

[8] Red Hat Customer Portal. Disponible en: https://access.redhat.com/

[9] Docs.openstack.org. *OpenStack Docs: Queens.* Disponible en: https://docs.openstack.org/queens/index.html

[10] Goyalankit.com. *Linux Bridge - how it Works.* Disponible en: https://goyalankit.com/blog/linux-bridge

[11] Ibm.com. *IBM Support.* Disponible en: https://www.ibm.com/support/home/

[12] Assaf Muller. Disponible en: https://assafmuller.com/

[13] Fiber Transceiver Solution. *OpenvSwitch vs OpenFlow: What Are They, What's Their Relationship?.* Disponible en: http://www.fiber-optic-transceiver-module.com/openvswitch-vs-openflow-what-are-they-whats-their-relationship.html

[14] Blogs.igalia.com. *Network namespaces - Unweaving the web.* Disponible en: https://blogs.igalia.com/dpino/2016/04/10/network-namespaces/

[15] Redhat.com. *The world's open source leader.* Disponible en: https://www.redhat.com/en